2012-07-09

A Server-Based Tool for Automating MODFLOW Simulations for Well Permitting Decision Support

David J. Jones
Brigham Young University - Provo

A Server-Based Tool for Automating MODFLOW Simulations for Well Permitting Decision Support

David J. Jones

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Civil and Environmental Engineering

Brigham Young University

August 2012

ABSTRACT

A Server-Based Tool for Automating MODFLOW Simulations for
Well Permitting Decision Support

David J. Jones
Department of Civil and Environmental Engineering, BYU
Master of Science

Numeric groundwater modeling techniques can assist water resources regulators pursuing prudent and foresightful aquifer management decisions. Unfortunately, the amount of time and professional expertise required to wield modern groundwater models often exceeds the resources of regulating agencies – even for simple modeling tasks that are repetitive in nature. In an effort to increase the accessibility of groundwater modeling resources, a server-based automated well permitting decision support system was designed. The prototype system allows a user to 1) input properties for any number of candidate wells, 2) execute an associated MOFLOW model, and 3) view relevant results of the simulation on a map such as drawdown contours and regions of decreased spring flow.

The system extends the existing concept of automated well permitting geoprocessing, which involves customizable tools built with ArcGIS and Arc Hydro Groundwater geoprocessing components, by moving the geoprocessing tool to a server and creating an interactive web interface built with the Google Earth plug-in. Several strategies to initiate the server-based geoprocessing tool were considered, with and without ArcGIS Server software. A realistic case study was included to demonstrate the system in action. Such server-based automated decision support systems have promising potential to increase the accessibility of groundwater models, facilitating professional management of crucial water resources.

Keywords: well permitting, decision support, geoprocessing service, MODFLOW

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Judicious management of groundwater resources depends on reliable and meaningful information. To evaluate groundwater extraction permits, regulating agencies require information forecasting the long-term impacts caused by the change in well pumping. While standard groundwater models can be used to obtain this information, the process can be costly and time consuming. New methods that streamline the well permitting process could help these agencies make better decisions with limited budgets.

In Utah, the Utah Division of Water Rights (UT-DWR) is responsible for processing applications for groundwater withdrawal permits. The process currently used by UT-DWR to analyze long-term aquifer impact due to candidate wells involves the modification of MODFLOW models developed by the USGS. With software applications such as GMS, the MODFLOW model for the relevant aquifer is opened and then edited to include new wells or to reflect any other changes that would result from application approval. Since such analyses are not an every-day task, UT-DWR has experienced difficulty maintaining a staff with the expertise necessary to perform this analysis.

The objective of this thesis is to create a web application for UT-DWR that will automate the well permitting analysis, making it possible for staff members without familiarity with groundwater modeling techniques to process applications for subsurface water extraction rights.

1.1 **Literature Review**

The significant number of scholarly efforts aimed at improving water resource data availability point to the relevant nature of such work. Many of these efforts incorporate the latest developments in service techniques and relational database models that allow users to access geographic information with a simple web-based map application. The Consortium of Universities for the Advancement of Hydrologic Science, Inc. has been at the forefront of recent collaborated academic efforts to facilitate hydrologic data sharing, providing standards for database design [1] and web services [2]. This endeavor has also included efforts aimed at sharing water models [3].

Web-based decision support systems have been widely discussed in the literature. Molenaar and Songer [4] identified several advantages of web-based decision support systems over traditional decision support systems for civil engineering applications, including increased accessibility (since users require only a browser and an internet connection) and increased centralization of system software (where modifications to the system must only be applied to the servers). More recently, decision support web applications have been enhanced with GIS-based web map technologies and applied to a variety of water resources management applications [5].

Several decision support systems for subsurface water management have been proposed, both web-based and otherwise. One off-line approach demonstrated by a case study incorporates monthly aquifer data to continuously calibrate and refine the model predictions to match the measured values [6]. Web-based decision support systems for groundwater management have traditionally been limited to data exploration and have not included modeling capabilities. One such example is an open source web GIS system, using a PostgreSQL database and web map application built with the standard Open Geospatial Consortium (OGC) web mapping services

(WMS, WFS, and WCS), that was used to improve groundwater management in Morocco [7]. Another web-based information system has been built by the USGS to make historic and current Arizona groundwater condition data available to decision makers and the public [8].

Jones, Strassberg, and Lemon developed a decision support system for well permitting analyses using a MODFLOW model [9]. The system uses the products of Strassberg's dissertation, the Arc Hydro Groundwater (AHGW) database design and geoprocessing tools [10] to encapsulate a MODFLOW model within the ArcGIS environment for pre- and post-processing. This transformation facilitates the creation of automated well permitting geoprocessing workflows that may be crafted to produce relevant map layers, perhaps polygons representing areas where drawdown exceeds a given threshold.

1.2 **Unique Contribution**

The work described in this thesis builds on the AHGW automated well-permitting methods developed by Jones, Strassberg, and Lemon. It describes the design of a web-based decision support system using the AHGW tools, making the system more accessible and even somewhat more automated. These efforts will enable groundwater resource managers to run simple MODFLOW simulations without any specialized software from an interface viewed in any common internet browser. The advantages common to all web-based decision support systems will become applicable to the realm of groundwater management.

1.3 **Objectives and Scope**

The server based system for automated well permitting was designed to meet the needs of UT-DWR through the following capabilities:

- The system should be maintainable and changeable.

- The tool should be capable of processing complex applications, such as those involving several wells.
- The web interface should be simple and easy to use while allowing the user to explore other relevant geographic information together with simulation results.

The Northern Utah County groundwater model [11] was selected as a pilot model for the server based system. It is a steady state model with multiple layers. Drawdown and change in spring flow were identified as the desired results for the tool to output. In accordance with UT-DWR practice, the tool does not immediately modify the MODFLOW model when groundwater applications are approved. Transient models, aquifers that necessitate other output geometries, and situations that require updating the model could be accounted for with similar designs, but were not considered in this study.

2 SYSTEM DESIGN

For maximized usability, utility, and flexibility of water resource geoprocessing services, Díaz *et al.* [12] recommended a layered approach where the distinct elements of the application – the data, presentation, and analysis elements – all work together but can also be accessed individually. The automated well permitting system may similarly be divided into three layers, with the MODFLOW model as the data element, the geoprocessing tool as the analysis element, and the web application as the presentation element. The foundational layer is the MODFLOW model. The middle layer is a geoprocessing tool that automates the MODFLOW analysis. This thesis is primarily concerned with the creation of the third layer, a web application that offers increased accessibility by serving the geoprocessing tool. Figure 2-1 shows the data flow diagram for the basic system design.

This layered components approach preserves the independent functionality of lower layers. This improves sustainability; if problems arise with the use of the web application, for example, the geoprocessing tool will likely remain functional and of some value until the web application can be repaired.

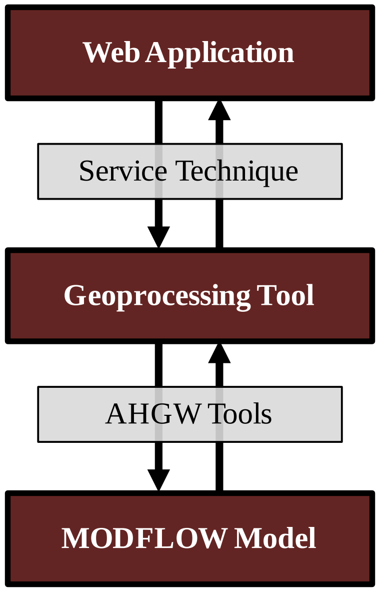


Figure 2-1. Simplified Data Flow Diagram for the Well Permitting Tool.

The MODFLOW model, geoprocessing tool, and web application are described in this chapter. The database used to input the candidate wells is also described. The next chapter discusses and compares methods to access the server-based geoprocessing tool over a network.

2.1 MODFLOW Groundwater Model

The foundational system layer is a MODFLOW groundwater simulation model. The prototype application is built to automate the Northern Utah County model, a model created and recently updated by the USGS [11].

MODFLOW simulations involve several files. Input files effect the simulation, and output files store the results of the simulation. The WEL file is typically the only input file that needs to be modified to perform a well permitting analysis.

UT-DWR uses several USGS MODFLOW groundwater models, each covering different areas, to evaluate groundwater extraction applications. Although software such as GMS may be used to open a model, modify the WEL file, execute the model, and then visualize the results of

MODFLOW simulations, this manual well permitting analysis requires technical expertise that may not be readily available. The ArcGIS geoprocessing tool automates this process in a customizable way.

2.2 **MySQL Database**

Before the geoprocessing tool is discussed, the approach used to input candidate well data will be described. The geoprocessing tool requires information about the candidate wells that will be analyzed; specifically, the well coordinates, screen top and bottom elevations, and pumping rate must be supplied by the user. To allow a user of the web application to provide these values quickly and accurately, especially for an application involving multiple wells, a database is used. The database stores a table of candidate well records. As outlined in Figure 2-2, it includes fields for the latitude, longitude, screen top elevation, screen bottom elevation, and pumping rate. Since permitting applications often involve several wells, an Application ID field was also included to group together well records that belong to the same application and should be analyzed together. This database scheme simplifies the work required to use the tool from the web interface; the interface user must only enter the appropriate Application ID into the tool input forms – much less effort than entering an entire table of data. Given the user-supplied Application ID, the geoprocessing tool queries the database to retrieve the complete dataset.

Both the web application and the geoprocessing tool interact with the database. In short, the web interface allows a user to add and edit data in the table, while the geoprocessing tool essentially uses the table as an input data source. The system design chart including the database component is shown in Figure 2-3.

Well	
This database table contains the candidate wells attributes required by the automated well permitting tool. Unless the geoprocessing tool incorporates unit conversions, units should match MODFLOW model units.	
Field Name	***Description***
WellID (key)	Uniquely identifies each candidate well record
ApplicationID	Identifies the water rights application associated with the well; multiple candidate wells may be integrated into a single water rights application
Latitude	Latitude in decimal degrees
Longitude	Longitude in decimal degrees
Flow	Rate of water diversion to be simulated by MODFLOW
ScreenTopElev	Greatest sub-surface elevation at the well location where water is transferred from aquifer to well; used to determine the model layer to place the MODFLOW well feature
ScreenBotmElev	Least sub-surface elevation at the well location where water is transferred from aquifer to well; used to determine the model layer to place the MODFLOW well feature

Figure 2-2. Description of Required Fields in the Well Table.

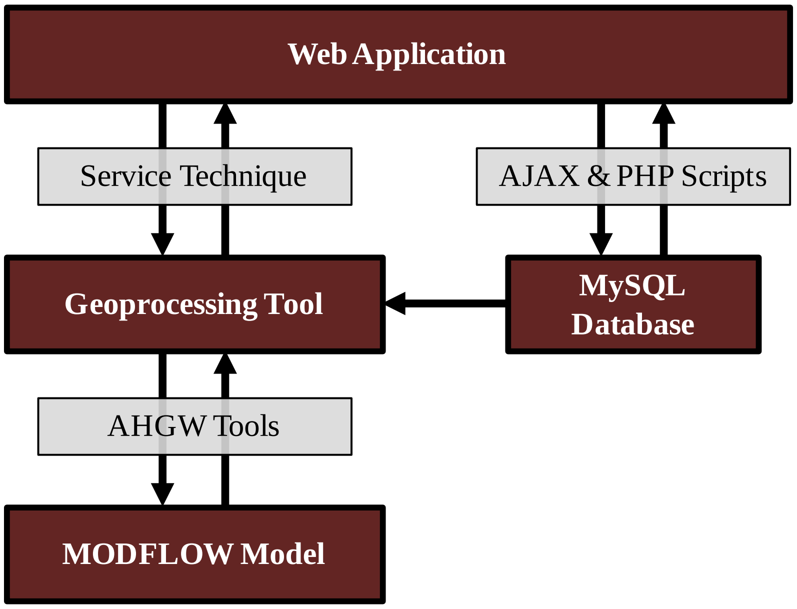


Figure 2-3. Data Flow Diagram Including the MySQL Database.

MySQL was selected as the database server because it can easily be accessed with PHP and is available for free. Unfortunately, MySQL is not officially supported by ArcGIS, the software platform the geoprocessing tool is based upon. To allow the ArcGIS-based geoprocessing tool to query records from the MySQL database, a custom script is used. Other database software that is supported by ArcGIS could be utilized without this complication.

2.3 **Arc Hydro Groundwater Tools**

As previously mentioned, the Arc Hydro Groundwater tools allow a MODFLOW model to be transformed to a geodatabase for use in the ArcGIS environment. The AHGW data model provides a standard format for hydrogeology data to be stored within geographic information systems. It is an extension to the surface water-oriented Arc Hydro Data Model. The book Arc Hydro Groundwater: GIS for Hydrogeology [13] summarizes the data model and outlines a set of geoprocessing tools that have been developed to incorporate the MODFLOW groundwater model within a GIS environment. Tools for converting between MODFLOW input and output files and an ArcGIS geodatabase allow MODFLOW models to be visualized within ESRI's ArcGIS applications. An *Execute MODFLOW* tool was also included, making it possible for a core set of groundwater modeling tasks to be executed exclusively within a GIS environment. The well permitting geoprocessing tool incorporates many of these AHGW tools in sequence to automate the basic groundwater modeling process required for a well permitting analysis.

2.4 **Geoprocessing Tool**

The geoprocessing tool automates the work required to modify, execute, and interpret the results of the MODFLOW simulation. The work is performed in an ArcGIS environment. Although expensive, ArcGIS software is commonly used in practice. AHGW tools allow ArcGIS to behave as a pre- and post-processer for MODFLOW. Utilizing this configuration, custom tools can be developed to automate the well permitting process as described by Jones, Strassberg, and Lemon. Because the geoprocessing workflow makes the server-based application possible, the following description is provided. Modifications required to facilitate server-based automation will be highlighted.

2.4.1 Custom ArcGIS Tools Created with ModelBuilder or Python

Some background information and definitions will be presented before the sequence of the geoprocessing operations is discussed. ModelBuilder is an ArcGIS application that may be used to connect a sequence of geoprocessing operations. It essentially creates customizable geoprocessing macros. A sequence of geoprocessing operations defined with ModelBuilder is sometimes referred to as a geoprocessing model, workflow, or custom tool since ModelBuilder workflows can be used in the same manner as the standard ArcGIS geoprocessing tools. The term “model” will be reserved for groundwater models rather than custom geoprocessing, the term “tool” will be used to refer to the black-box process that generates desired output from provided inputs, and the term “workflow” will be used when discussing the inner workings and logic of the custom tool. Terms such as geoprocessing service and geoprocessing task have specific meanings associated with ArcGIS Server; when an ArcGIS toolbox is published with ArcGIS Server, the toolbox becomes a geoprocessing service, and each contained tool becomes a geoprocessing task.

Python scripts provide an alternative to ModelBuilder workflows. Custom tools created with ModelBuilder can only be used from other ArcGIS software like ArcMap or ArcCatalog. Scripts created with Python are more flexible; they may be designed to function within an ArcGIS application as custom script tools, behaving exactly like a ModelBuilder tool, or as stand-alone geoprocessing scripts designed to run outside a standard ArcGIS application, perhaps accessed from the command line. With little effort, a single script may be designed to function as either a script tool or a stand-alone script.

The relative advantages and distinctions between a ModelBuilder workflow, a Python script tool, and a stand-alone Python script will be discussed in the following chapter with various techniques for serving the tool. The sequence of geoprocessing operations is the same for the well permitting tool, whether it is constructed with ModelBuilder or Python.

The well-permitting workflow may be divided into four steps: 1) modify the MODFLOW input files to account for the candidate wells, 2) run MODFLOW, 3) convert the MODFLOW output files to resultant GIS datasets, and 4) for each resultant dataset, produce KMZ or PDF result files that are specially formatted and styled to effectively convey meaningful information to the user. The fundamental geoprocessing functions and tools for each of these steps are shown in Figure 2-4.

2.4.2 Part 1 of the Workflow: Modify the WEL File

The MODFLOW WEL file must be modified to include new records for each of the candidate wells before the simulation is run. The WEL file simply lists the pumping rate and the I, J, K indices of the cell containing each well for each time-step. The first task of the geoprocessing tool is to convert the coordinates and screen elevations of each candidate well into I, J, K values and to write a new WEL file.

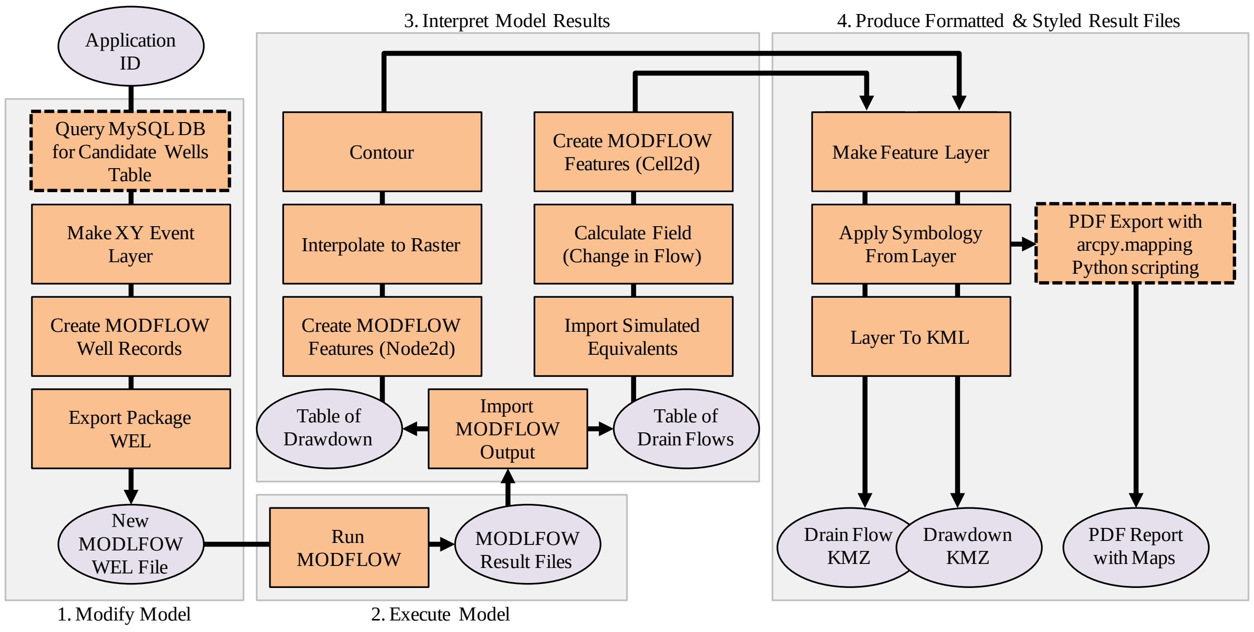

Application ID
1. Modify Model
Query MySQL DB for Candidate Wells Table
Make XY Event Layer
Create MODFLOW Well Records
Export Package WEL
New MODLFOW WEL File
2. Execute Model
Run MODFLOW
MODLFOW Result Files
3. Interpret Model Results
Contour
Interpolate to Raster
Create MODFLOW Features (Node2d)
Create MODFLOW Features (Cell2d)
Calculate Field (Change in Flow)
Import Simulated Equivalents
Table of Drawdown
Import MODFLOW Output
Table of Drain Flows
4. Produce Formatted & Styled Result Files
Make Feature Layer
Apply Symbology From Layer
Layer To KML
PDF Export with arcpy.mapping Python scripting
Drain Flow KMZ
Drawdown KMZ
PDF Report with Maps

Figure 2-4. Simplified Geoprocessing Workflow for the MODFLOW Well Permitting Analysis.

The workflow starts with the Application ID provided from the user. A custom tool is used to generate a table of candidate wells from this ID. This custom Python script tool uses the *urllib* module to open and read the contents of PHP page, the same PHP page that is used by the web application to query and add candidate wells to the map for a specific Application ID. The query results, including the coordinates, pumping rate, and screen elevations for each candidate well, are saved to an ArcGIS geodatabase table. While it is possible to use Python to query a MySQL database directly, this functionality is only available after installing a custom module such as *MySQLdb*. Using the *urllib* module allows the Python script to be transferred to another machine and work without requiring a non-trivial Python module installation. A well permitting tool built with an ArcGIS-supported database would use the *Table Select* tool rather than the custom Python script to do this query. In both cases, the Application ID is used to form a query and a table of the candidate wells of interest is returned with the latitude, longitude, pumping rate, and screen top and bottom elevations of each candidate well.

The next step is to convert the candidate well table into a map layer. This is accomplished with the *Make XY Event Layer* tool using the latitude and longitude values stored in the table. Then the *Feature Class to Feature Class* tool is used to convert the layer to a feature class and also to update the spatial reference. Subsequent AHGW tools require the spatial reference of each input dataset – the coordinate system, resolution, and tolerance – to match each other. The environment settings for the *Feature Class to Feature Class* tool were set to modify the spatial reference, causing the coordinate system to be projected from geographic to Utah State Plane.

With a feature class containing the candidate wells, the workflow can proceed in the same manner described by Jones, Strassberg, and Lemon using AHGW tools. *Create MODFLOW Well Records* appends the candidate well records to a table of all MODFLOW well features, and *Export Package WE*L writes out the WEL file using this amalgamated table. With the newly generated WEL file, the MODFLOW simulation is ready to be executed.

2.4.3 Part 2: Executing MODFLOW

The execution of the MODFLOW simulation is performed with a single AHGW tool, *Run MODFLOW*. This tool requires two input parameters: the path to the MODFLOW name file, and the path to the executable. A MODFLOW executable file is downloaded with the AHGW tools and may be copied into a common directory, accessible with a relative path. If the workflow is created in ModelBuilder, a precondition must be created to start the *Run MODFLOW* tool only after the new WEL file is written.

2.4.4 Part 3: Creating Output KMZ Files

For the Northern Utah County model, aquifer drawdown and the change in spring discharge were identified as the results of interest to have the geoprocessing tool calculate. The AHGW tools facilitate obtaining these results from the MODFLOW result files.

The *Import MODFLOW Output* tool populates tables with the calculated drawdown for each grid. The *Create MODFLOW Features* tool then joins this table with point geometries representing the cell center of a single MODFLOW layer. A drawdown raster is then generated with the *Natural Neighbor* tool, from which contour lines are produced with the *Contour* tool.

To obtain the change in spring flows, the workflow first uses the *Import Simulated Equivalents* tool to populate the FLOB table with the simulated drain flows. Although the FLOB table is meant to store flow observation data for parameter estimation runs, it may be pre-populated with the baseline flows rather than flow observations. After the *Import Simulated Equivalents* tool populates the observed flow field, the *Calculate Field* tool is used to calculate the residual flow field which really represents the desired change in flow. *Join Field* then appends the MODFLOW K values which indicate which layer each drain belongs to. The next tool, *Create MODFLOW Features*, turns the table into a geographic feature class with rectangular MODFLOW cell polygons for each drain element.

2.4.5 Part 4: Applying Symbology

Web interface users are not able to customize the appearance of geoprocessing results. This makes it important for newly created geoprocessing results to be delivered with useful symbology. This requirement may be satisfied through additions to the workflow. The techniques that will be described are applicable to both result KMZ files and PDF maps, though PDF output is not discussed until Section 3.3.

2.4.5.1 Applying Symbology to Single Independent Layers

The general procedure used to create deliberately formatted KMZ output files is as follows:

1. The result feature class is converted to a layer with *Make Feature Layer*.
2. A predefined symbology is assigned to the new layer using *Apply Symbology From Layer*.
3. Finally, the *Layer to KML* tool creates a nicely formatted KMZ representation of the results.

The *Make Feature Layer* tool is required because only layer files, not feature classes, can be converted to KML or KMZ files with the standard ArcGIS conversion tool. A few of the enhanced geoprocessing outputs listed in Section 3.3.2 also require the geographic results to store as a layer.

Predefined symbologies are layer files created in ArcMap. Using such layer files, the *Apply Symbology From Layer* tool can be used to apply a graduated color ramp to features based on values in the attribute table. The color ramp classification can either be held constant or adjusted to fit the distribution of attribute values for the target feature layer. When the symbologies template layer is made with a manual classification, the color ramp classification is held constant; conversely, if a classification method like Equal Interval or Natural Breaks is used, the color scale will update dynamically to fit the new range of values. The template layer file can also be used to define other properties that will impact the usability of the final KMZ result file. When a user clicks on a feature from a KML or KMZ file, an info-balloon displays additional information. This content can be controlled to some degree using the symbology template layer file.

Finally, the *Layer to KML* tool converts the layer to a KMZ file. KMZ files are simply zip archives that combine a KML file with icon images or other data required for proper display. KML and KMZ files can be interpreted by many clients, including the Google Earth plug-in.

2.4.5.2 Applying Matching Symbology Across MODFLOW Layers

In some cases, the *Apply Symbology From Layer* geoprocessing tool must be supplemented with additional tools in order to achieve the desired results. One significant case that requires consideration is when the well permitting simulation must create different KMZ output files for the same variable of interest but for different MODFLOW layers. The color scale should match across layers. In most cases, however, the different MODFLOW layers will have different attribute value ranges, and this will cause the color scale to be different for each layer.

The change in spring flow output serves as an example of the issue. The North Utah County MODFLOW model has drain features in layers 1, 2, and 3. Because many of the drains in different MODFLOW layers have the same I and J coordinates, a convenient way to display the results is to produce 3 different KMZ files, with separate files for layer 1, 2, and 3. The color red might be used to signify drains with the maximum flow changes, but the minimum and maximum flow change will almost certainly be different for each layer, and it would be preferable for the color red to mean the same thing regardless of MODFLOW layer. This challenge is overcome by combining the features for each MODFLOW layer, then using the *Apply Symbology From Layer* tool, and finally splitting the results into separate MODFLOW layers. The following operation achieves consistent formatting of multiple layer change in spring flows output:

1. Create MODFLOW Features from a derivative of the FLOB table with no k value specified. (The FLOB table is copied and then joined with the FLOBFactors table

to get the K field, the MODFLOW layer identifier necessary for operation 5 below.)

2. Copy Features
3. Make Feature Layer
4. Apply Symbology From Layer
5. Select Layer By Attribute
6. Delete Features
7. Layer To KML
8. Repeat 2 – 7 for each MODFLOW layer

A similar procedure could be used to match contour line symbology across layers.

2.5 **Web Interface**

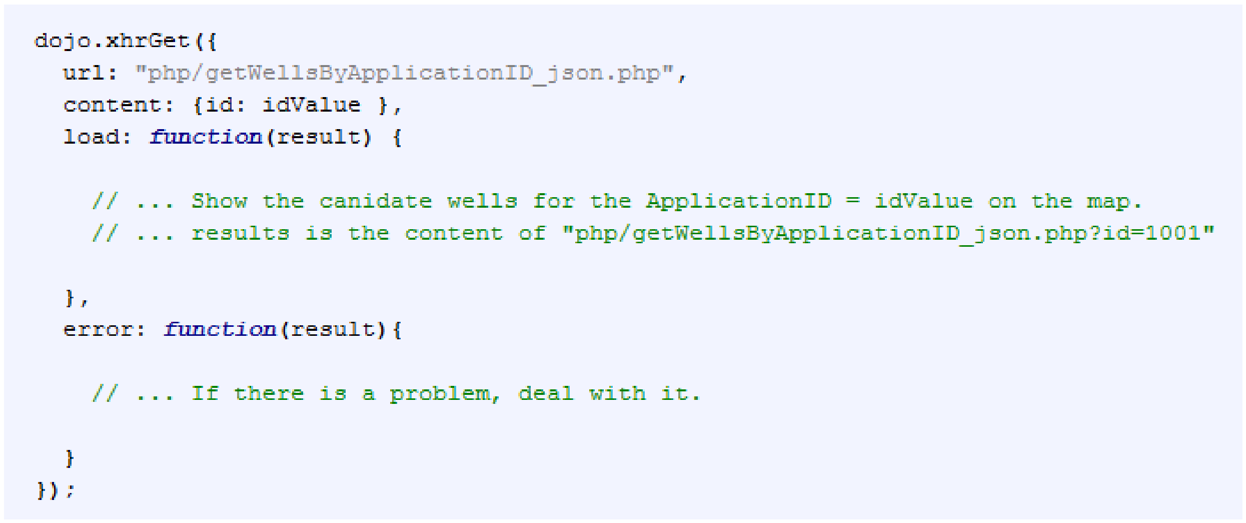
The role of the web application is to provide simplified access to the well permitting geoprocessing tool. In short, a set of form input boxes and a submit button allow the user to execute the tool. While the server executes the tool, status messages may be updated and displayed on the web interface. An interactive embedded map displays the final geographic results. Since the tool input refers to values stored in a database table, the web application also allows a user to view, modify, and add these database records. The web interface is HTML enhanced with JavaScript and CSS. The Dojo JavaScript Toolkit and the Google Earth plug-in are leveraged to increase usability.

2.5.1 Dojo JavaScript Toolkit

JavaScript is a client-side scripting language that makes HTML web pages more dynamic. The Dojo Toolkit [17] is collection of high-level JavaScript tools. These tools are used by the web interface to communicate in the background with the server, format the page with specialized widgets, and ensure the application works the same in most browsers. There are many other capable JavaScript libraries, and Dojo was initially selected because it is required by the ArcGIS Server JavaScript API; however, the ArcGIS API ultimately was not used for any of the service strategies described in Chapter 3.

The most valuable contribution of the Dojo JavaScript Toolkit is the tool that facilitates background communication with the server, the *dojo.xhrGet()* function. Rather than redirecting the user to a new page to display the results of the tool, this function allows the web application to send the job request to the server and retrieve the results in the background. This web development technique, sending and receiving information without reloading the page, is known as Ajax. While the Dojo Toolkit is not required to use Ajax techniques, Dojo provides a convenient tool that makes Ajax requests easier. (In fact, the Google Earth plug-in has its own Ajax-based tool for loading KML or KMZ files in the background.)

For example, consider the code shown in Figure 2-5. When a user enters an Application ID, the web interface should display any candidate wells with this ID on the map. To accomplish this, the web application creates an Ajax request for the coordinates of candidate wells with a specific Application ID. The URL of the resource, along with Application ID value that will be appended to the URL to form a GET request, are passed into the *dojo.xhr.Get()* function. Callback functions to complete when the resource is loaded or when it fails to load are also specified. In this example, the resource is a PHP script that queries the MySQL database and prints out coordinate information for candidate wells with the specified Application ID. Rather than displaying the sets of coordinates as raw text in the browser, the text is stored by the *results* variable and passed into the *load* function where it is then used to add well placemarks to the map.



```
dojo.xhrGet({
  url: "php/getWellsByApplicationID_json.php",
  content: {id: idValue },
  load: function(result) {

    // ... Show the canidate wells for the ApplicationID = idValue on the map.
    // ... results is the content of "php/getWellsByApplicationID_json.php?id=1001"

  },
  error: function(result){

    // ... If there is a problem, deal with it.

  }
});
```

Figure 2-5. Example Usage of the dojo.xhrGet Function.

The web interface uses background Ajax requests extensively. All dynamic data is loaded into the web interface in this manner. In most cases, the application uses specifically designed PHP pages as the resource so the returned data can be dependent on arguments passed along using the GET method; however, the resource may be any file type. The application employs Ajax techniques not only to retrieve data, but also to initiate specific tasks – such as modifying the database or executing a Python script – where the returned data would primarily indicate whether the task was completed successfully.

Another use of the Dojo Toolkit is to enhance visual appeal and usability. A Combo Box widget is used to create an advanced text box for the Application ID. It includes an auto-complete dropdown menu containing valid Application IDs. Dojo formatting widgets are used to organize the web page and to show and hide content. Cascading Style Sheets are used to control the display of Dojo widgets and other web page elements in the application. Dojo uses a predefined style sheet that can be modified or overridden.

Title pane widgets, such as the example show in Figure 2-6, are used to generate collapsible DIV elements. A title above the collapsible content is always visible and may be clicked to show or hide the underlying content. Title panes were used by the prototype web application to hold the tool input form and to hold the list of optional map layers. Modified title pane widgets were embedded within each other to create this list of map layers. The user can toggle layers on and off, open and collapse sub-layers, and view legends for each layer. Dojo widgets may be modified simply by creating a new style class and specifying the desired formatting in a CSS file. Then the custom style class is assigned to the title pane HTML element with the class attribute. The custom title pane style eliminated the border, background, and padding around the label; and includes only a dotted line border on the left side of the contents. Figure 2-7 shows an example. A display similar to the Google Earth "Layers" pane or the ArcMap "Table of Contents" was the goal. Title panes allow content to be organized and condensed, reducing the required display area.

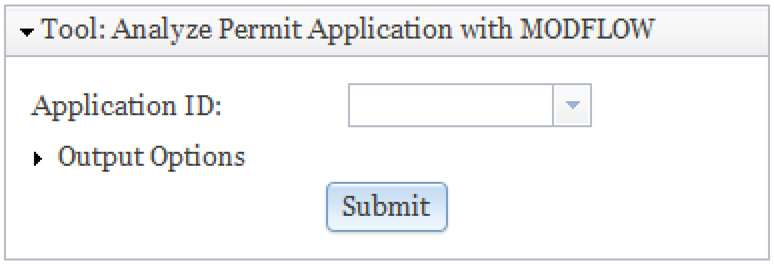


Figure 2-6. A Standard Title Pane Expanded to Show Tool Input Forms.

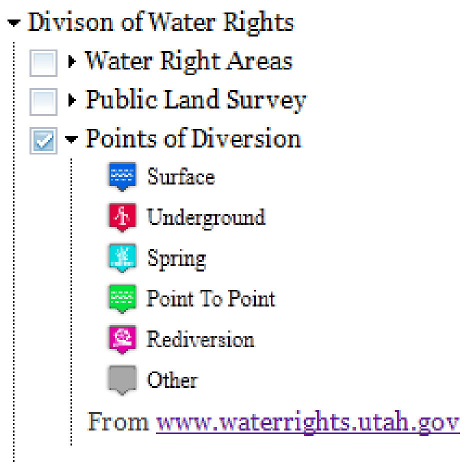


Figure 2-7. Custom Title Panes Embedded within the Contents of Other Title Panes to Form a Table of Contents for Map Layers.

One final benefit from the Dojo Toolkit is enhanced cross-browser compatibility. Users expect web applications to function identically on all major browsers. Dojo reduces the effort and technical expertise required to meet this demand.

2.5.2 Google Earth Plug-in

The Google Earth plug-in is another free JavaScript extension. An embedded Google Earth map is used to display the locations of candidate wells, the output from the geoprocessing task, and other KML and KMZ files that enhance the well permitting process. The Google Earth map interface has many benefits over other web mapping options. It responds quickly to user interaction and includes useful features such as 3D terrain, street-view imagery, and historic imagery. The display of these components and layers may be toggled on and off. A disadvantage is that the end user must download and install a plug-in, and that it is not supported by all browsers and operating systems.

A critical capability of the Google Earth plug-in is the ability to add KML and KMZ files directly to the map. This feature is leveraged by the application to display tool results. The result of the geoprocessing tool is a series of KMZ files; after the tool completes, it send back the URLs of these KMZ files, and they can be added to the map. Dynamically-added check boxes allow these KMZ layers to be toggled on or off. Without the capability of importing KMZ files, the geoprocessing tool would have to convert the results to a table of coordinates, and the application would have to convert these coordinates back to geographic features. KMZ file are preferable to coordinate output not only because it simplifies the web application source code, but also because it is a widely recognized file format supported by several applications.

The web application also allows the user to browse pre-created KML and KMZ layers that display a variety of useful information. KML layers displaying inputs to the MODFLOW model were created with AHGW tools and are listed as optional layers that a user may toggle on and off. These MODFLOW layers include ground overlay images showing the model active grid, baseline head, and hydraulic conductivity. KML point and polygon features were created to represent MODFLOW sources and sinks such as wells, drains, evapotranspiration, streams, and recharge. The opacity of each individual layer may be adjusted. The Google Earth plug-in allows layers to be grouped into folders, and the opacity of folders may be adjusted as well.

UT-DWR has a collection of KML files with pertinent water rights information. One such KML file contains a network link that displays the locations of individual water rights, each including an info balloon with a link to more information. This KML file and others were added as optional layers to the application. With results from the tool and these KML network links, it is possible to visualize the specific water rights that will be affected by candidate wells.

Other embeddable map interfaces were considered. The Google Maps JavaScript API has many of the same benefits of the Google Earth API and is more widely used by web developers. It also allows KML and KMZ files to be directly added to the display, but the current version limits the size of the KML file, the number of network links, and the total number of features that may be in memory at a time [14]. As a result, it may not be possible for an application built with the Maps API to load medium to large results from the geoprocessing tool or to display all the available map layers simultaneously. ESRI has developed an extension to the Maps API that helps developers display geoprocessing service result geometries on an embedded Google Map; however, this extension was built for the outdated version 2 of the Maps API. Further, the extension only works with ArcGIS geoprocessing services. ESRI created this extension using the Ajax tools of the Dojo toolkit, tools that were used to add the same functionality to the Google Earth API based application being described and could be applied to version 3 of the Maps API.

2.5.3 Database Interaction

The web application allows multiple avenues of interaction with the candidate well records from the MySQL database. When an Application ID is entered in the input text box, the map automatically displays the locations of all the wells associated with this application. The data stored for each well is displayed in an info-balloon when a particular well is clicked by the user. An optional layer allows the user to view the well records for all Application IDs.

The web application also displays the database in tabular form, allowing read and write access. Ajax techniques are used to submit PHP scripts that retrieve records for display or modify a record. Separate PHP scripts for editing, adding, and deleting records were created.

3 SERVICE STRATEGIES

The geoprocessing service must at minimum allow a user of the well permitting application to perform two fundamental functions:

1. Initiate the remote execution of the geoprocessing tool with supplied arguments, and
2. Access the results.

This chapter discusses several strategies that may be employed to provide these functions. Two methods for initiating a geoprocessing service will be described. The first initiation method uses ArcGIS Server software, and the second uses a PHP and Python solution. Two possible result retrieval paradigms are described for the PHP and Python solution. The basic retrieval paradigm is appropriate for tradition ArcGIS geoprocessing tools for which the nature of the expected output is pre-defined; an advanced retrieval paradigm that allows more flexibility and interaction between the geoprocessing tool and the web interface is also described. In all, three geoprocessing service strategies will be described, each with distinct advantages.

3.1 **Basic ArcGIS Server Strategy**

ArcGIS Server is software that can be used to share geographic data across a network. One way it facilitates data sharing is with web services. Geoprocessing web services can be created to share geoprocessing capabilities over a network. Conventionally, a published geoprocessing toolbox is called a service, and each tool inside the toolbox is called a task. Any developed application can be programmed to utilize an ArcGIS web service. Published services can also be utilized within applications like ArcMap and ArcGIS Explorer.

3.1.1 Implementation with System Components

Figure 3-1 shows the system design for an application built around an ArcGIS geoprocessing service. (In contrast to the basic data flow diagrams of Chapter 2, the service technique is displayed as an element rather than a transitional method in order to show added detail.) The ArcGIS Server implementation requires the geoprocessing tool to be converted to a geoprocessing task and the web application to be designed to communicate with the ArcGIS Server REST Interface. Also, the server must be configured in a way that will satisfy the AHGW tools license manager. These issues are discussed in the following sections.

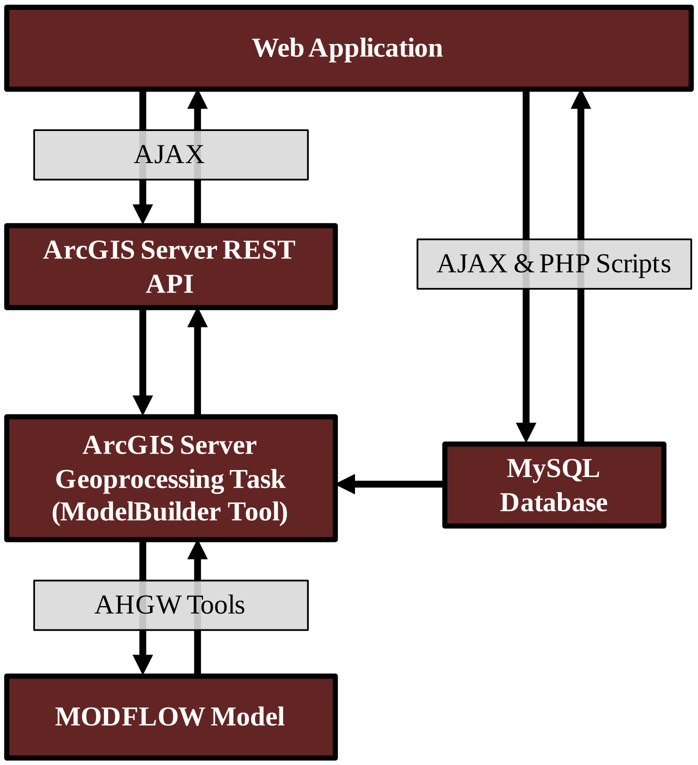


Figure 3-1. Data Flow Diagram for the ArcGIS Server Geoprocessing Task Method.

3.1.1.1 ArcGIS Server REST Interface

ArcGIS Server web services allow client applications to access resources through a REST interface. This interface allows clients of the well permitting geoprocessing task to initiate a MODFLOW simulation by accessing a URL resource, and to retrieve the results of the simulation using other URLs as defined in the ArcGIS Server REST API [15]. Table 3-1 summarizes the available URLs from the REST API specifically for asynchronous geoprocessing tasks.

Table 3-1. List of ArcGIS Server REST URLs for Asynchronous Geoprocessing Tasks

Service Resource	**URL**
Access task metadata	http://<gpservice-url>/<taskName>
Submit a job	http://<gpservice-url>/<taskName>/submitJob
Get job status and messages	http://<gpservice-url>/<taskName>/job/<job-id>/results
Get job results	http://<gpservice-url>/<taskName>/job/<job-id>/results/<param-name>
Get job inputs	http://<gpservice-url>/<taskName>/job/<job-id>/inputs/<param-name>

All the geoprocessing service resources are completely accessible through this set of URLs. A client application tells the server to run a geoprocessing task by accessing the submit job URL appended with arguments to specify input parameters. Because it is an asynchronous geoprocessing task, the submit job URL does not return any results but rather returns a job-id text string. The client application is expected to check back periodically by accessing the results URL for the given job-id. When results are available, the client then accesses the result URL for each output parameters of interest. In depth documentation of the ArcGIS Server REST API for geoprocessing services is available from ESRI [15].

3.1.1.2 Server Configuration for AHGW Tools

Some AHGW tools are free to use without cost; others, including several used by the well permitting tool, require an AHGW license. Every time a licensed AHGW tool is executed, the tool checks to verify that the current user has obtained and configured an appropriate license from Aquaveo. ArcGIS Server uses a specific user account, named ArcGISSOC by default, to handle web services. In order for AHGW geoprocessing tools to function as part of a geoprocessing task, an AHGW license must be enabled for this ArcGISSOC user account. Whether AHGW tools are enabled for any other user is completely irrelevant to the functionality of the geoprocessing service.

Configuring a license simply involves running an executable file included with the AHGW tools download and specifying the network location where a network hardware lock resides. With the ArcGISSOC user account name and password, an administrator may log onto the server and enable the AHGW license in this manner.

Even after the AHGW license is properly configured, tools requiring an AHGW license have been found to behave poorly when used with an ArcGIS Server geoprocessing service. The first errand of each licensed tool, completing AHGW license verification, was identifies as taking an inordinate amount of time to complete – about a minute. Unfortunately, the cause of this poor performance was not identified. This unresolved issue significantly increases the time required for the geoprocessing tool to execute, diminishing the value and appeal of the application.

3.1.1.3 Geoprocessing Tool Requirements and Modifications

An ArcGIS Server geoprocessing task is built from a specially configured ArcGIS geoprocessing tool, either a ModelBuilder workflow or a Python script tool. Detailed configuration requirements for the tool configuration may be found in the ArcGIS Server documentation [16]. In short, input and output parameters must be formally defined, the *scratchworkspace* variable must be used to define the location of outputs, and intermediate data must be managed deliberately.

Because the purpose of many geoprocessing web services is to generate geographic features that can be displayed on a simple web map, and due to the fact that common ArcGIS file types such as shapefiles and feature classes within geodatabases are difficult for light-weight web applications to interpret, an ArcGIS Server geoprocessing service automatically converts output geographic features to a simplified data structure. The desired output data structure may be specified using the REST interface. Conveniently, KMZ file output is one of the possible output formats. This capability allows for a minor simplification to the geoprocessing workflow; the workflow may optionally omit the conversion processes performed by the web service: the conversion of the drawdown contour lines feature class to a layer file, the explicit *Apply Symbology from Layer* tool, and the explicit use of *Layer to KML*. If ModelBuilder is used, template symbology layer files should be assigned to each output parameter feature class to prevent arbitrary output symbology.

When ArcGIS Server is used to generate a geoprocessing service from a geoprocessing workflow, several configuration settings must be considered. For the automated well permitting tool, the following settings are recommended:

- **Asynchronous execution –** Because the tool takes more than a couple of seconds to complete, it should be configured as an asynchronous service rather than as a synchronous service. According to ArcGIS Server help documentation, synchronous geoprocessing services are not appropriate for time-intensive services that return large amounts of output data. The difference between synchronous and asynchronous services is illustrated by the fact that synchronous geoprocessing tasks require accessing a single REST interface URL that accepts input arguments, executes the tool, and returns all the output results. As a result, intermediate status messages cannot be provided by synchronous services.

- **Show messages enabled –** The web application uses these messages to provide status updates.
- **Maximum number of instances limited to one –** Errors due to simultaneous request to access and modify the same datasets can be avoided by limiting the number of service processes to one. ArcGIS Server will assign a "Waiting" status message to new geoprocessing tasks when an existing task is being executed. Upon completion of the existing task, the oldest waiting submission will be executed.

3.1.1.4 Web Application Modifications

The web application features that allow a user to submit a job, view status messages, and view the results each require interaction with the geoprocessing services REST interface. Although ESRI provides various APIs with these capabilities to developers for certain web mapping interfaces, a custom interface to interact with the geoprocessing service was created as part of this research. This custom interface was built using Dojo Ajax requests, the same method that is likely used by at least one of the APIs maintained by ESRI.

As discussed in Section 2.5.1, Ajax requests allow a user to access URL resources in the background. This makes it relatively easy to access the geoprocessing service REST framework to submit jobs, check and update the status, and display the results. A custom JavaScript object was created with these capabilities and used by the web application.

3.1.2 Summary

The basic ArcGIS Server strategy, using ArcGIS Server to initiate the geoprocessing tool and a custom implementation of the REST API to retrieve status messages and KMZ results, has advantages and limitations that will be discussed.

3.1.2.1 Advantages

ArcGIS Server provides several capabilities that enhance maintainability. The software comes with documentation, tutorials, and examples that could be used by anyone to learn the procedures for maintaining services. As mentioned previously, tools have been developed to facilitate the development of ArcGIS web map applications, applications that can make geoprocessing services and many other GIS resources available through a web browser. Although a custom web mapping solution was implemented rather than an ArcGIS web mapping solution in order to achieve project-specific objectives, it is possible that some organizations would give up the interactive Google Earth interface for web applications that are easier to create with ArcGIS Server. These generic web-map applications could back-up the primary web interface, ensuring some level of service in the event of unforeseen problems with the highly customized interface.

ArcGIS Server Software, although expensive, provides the legal permissions required to publish ArcGIS geoprocessing tools over the internet. While access over a local network may be sufficient for many situations, others may call for increased access to the application. When a Server license is required to satisfy legal permissions, the cost deterrents is eliminated and agencies may be more inclined to exploit the corresponding software.

3.1.2.2 Limitations

A critical drawback of this method is the excessive amount of time required by licensed AHGW tools to authenticate. Before committing to a serious implementation of this method, an organization would want to better understand this issue.

Another shortcoming of this basic ArcGIS Server method is the amount of effort required to customize the web interface to work with the REST API. Although this effort is comparable to the work required by the other methods that will be discussed, a common selling point for ArcGIS Server software is that customized web-map applications incorporating the services are easy to produce. As a result of this method’s custom web interface, professionals familiar with ArcGIS Server products may not necessarily have the background required to maintain the entire system as described.

3.2 **Basic PHP and Python Strategy**

Python and PHP may be used to make a custom light-weight geoprocessing service. This simplified method may be appropriate for making the server-based geoprocessing tool accessible over local intranets in a very simplified manner.

3.2.1 Implementation with System Components

Implementation of the simplified PHP/Python method necessitates additional system components and attention to the data flow methods. Figure 3-2 displays a summary of the resulting system.

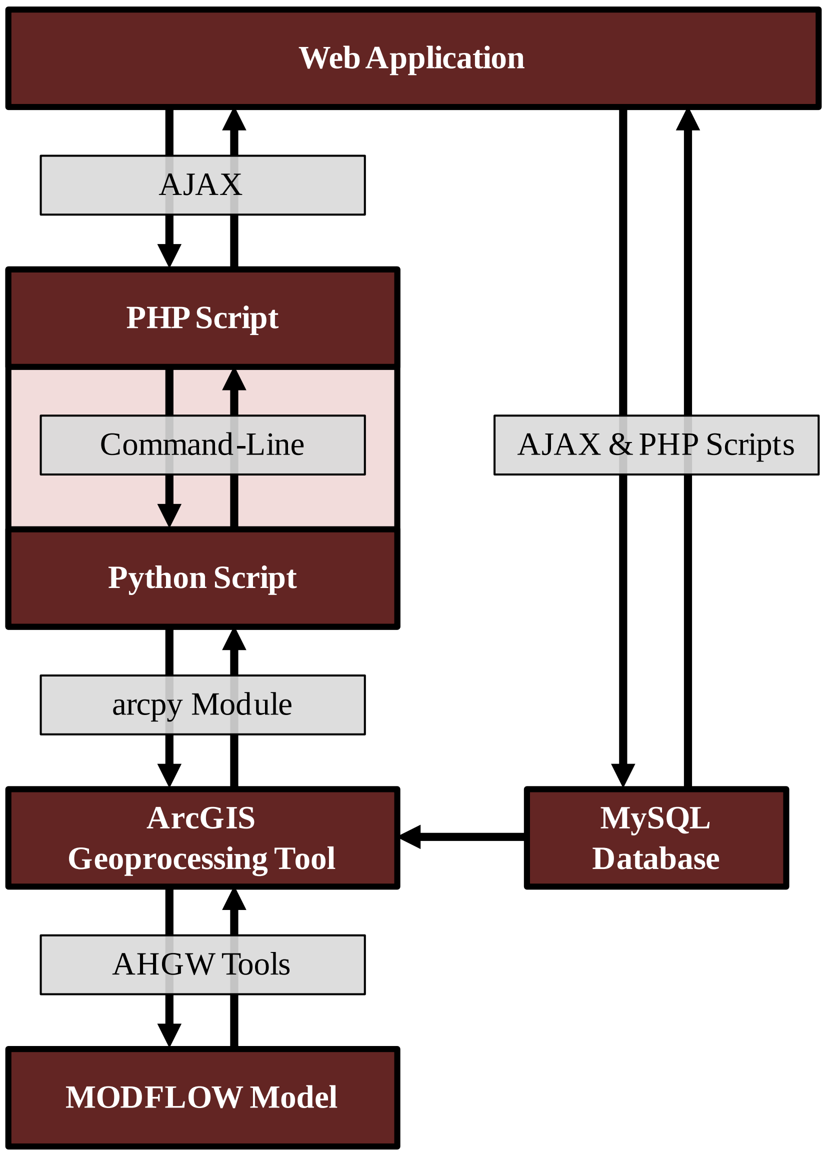


Figure 3-2. Data Flow Diagram for the Elementary PHP/Python Method.

The ArcGIS geoprocessing tool can be either a ModelBuilder workflow or Python script tool, as described in Section 2.4.1.

3.2.1.1 PHP Script and Python Script

The PHP and Python scripts play vital but limited roles in the simple method. Used in succession, their role is to remotely initiate the geoprocessing tool and return the results to the web application. The web application uses Dojo Ajax techniques to call the PHP script; the PHP script initiates the Python script; finally, the Python script initiates the geoprocessing tool.

The Python script is designed to run from the command shell. It uses the *arcpy* module to execute the ArcGIS geoprocessing tool built with ModelBuilder. The Python script takes arguments from the command line and passes them directly to the ArcGIS geoprocessing tool. When the tool concludes, the geoprocessing messages are returned. Adding this layer to the system allows the geoprocessing tool to be initiated in a non-ArcGIS environment.

PHP has standard functions that may be used to execute external programs, programs such as the Python script. The PHP script takes arguments supplied in its URL reference by the client and passes them directly to the Python script. When the Python script completes and returns the geoprocessing messages, these messages are printed out to the PHP page for viewing by the client. As a result, the PHP script layer accomplishes the main goal of making the geoprocessing tool accessible over a network with a URL.

There are likely alternative methods for publishing the geoprocessing tool. For example, it is possible to configure a server to serve Python scripts, possibly eliminating the need for the PHP script; however, PHP is already used by the web application for MySQL interaction and is a more popular service language than Python. While it may be possible for other programming languages to call the ArcGIS geoprocessing tool, Python is the preferred language for most automated ArcGIS applications due to the recent focus of ESRI development. The use of the combined PHP/Python approach is also favorable for adding the additional functionality of the more involved PHP/Python approach, discussed in Section 3.3.

3.2.1.2 Server Configuration

The AHGW license must be enabled for the account that remotely executes the geoprocessing tool. The solution for the ArcGIS Server method as described in Section 3.1.2 was simply to enable the AHGW license for the ArcGISSOC user account, the account ArcGIS Server utilizes to execute its tasks. A similar solution must be implemented for this alternative service method.

In IIS 7, anonymous authentication can be used to permit a client to access published resources without requesting a user name and password. A user account may be specified for the client to assume, defining the level of access that will be granted. This authentication scheme, with the built-in IUSR user account credentials, is the default configuration. A simple way to accommodate the AHGW licensing requirement is to use anonymous authentication and, rather than using the default IUSR account, to provide an anonymous authentication user account that has already enabled the AHGW tools. For unknown reasons, successful AHGW license validation also requires the user account to have a separate process running on the machine at the time of execution. A primitive means to satisfy this stipulation is to leave the user account continuously logged on. Alternatively, the ArcGISSOC account is a convenient choice for the anonymous authentication, provided ArcGIS Server is available, since it always has a process running when ArcGIS Server is enabled.

Because the PHP and Python scripts requires more time than normal web resources to load, further configuration may be necessary to prevent the server from timing out. For example, implementations with Windows Server may require adjustment of the CGI timeout variable and the FastCGI activity timeout and request timeout variables.

3.2.2 Summary

The basic PHP and Python strategy and the basic ArcGIS Server strategy both accomplish the same general objective of making an ArcGIS geoprocessing tool accessible over a network, but the respective implementation methods differ significantly. The advantages and disadvantages of the PHP and Python strategy will now be considered.

3.2.2.1 Advantages

The simplicity of the service configuration is the primary advantage of this service strategy. Although new PHP and Python elements are introduced, their role is simply to initiate the server-based geoprocessing tool; as a result, these scripts are relatively short and straightforward. Other system components are only minimally affected.

Interoperability is an important benefit stemming from the simplicity of the system. Unlike the ArcGIS Server method, the geoprocessing workflow does not require any specialized configurations unique to the geoprocessing service; rather, the same well permitting geoprocessing tool that is called by the PHP and Python scripts can be used “manually” with ArcMap. This simplifies the modification and testing procedures.

ArcGIS Server is expensive software that is designed to create robust web services. If all the high-end features of an ArcGIS geoprocessing service are not needed, a custom strategy may save resources. If the web permitting is to be limited to a private network rather than the world-wide internet, the application may comply with the ArcGIS concurrent-use license agreement, eliminating the need for an ArcGIS Server license.

3.2.2.2 Limitations

The simplified PHP and Python service strategy as described has a few limitations. It only supports a single user at any given time, and returns an error when used simultaneously. It also does not make progress or status messages available to the web interface; this limitation is addressed with the enhanced PHP and Python strategy, though the approach introduces new limitations and trade-offs.

The capacity to store previous simulation results is also lacking. ArcGIS Server stores past geoprocessing results for a configurable amount of time. The basic PHP and Python method, on the other hand, overwrites the previous result files with each execution. This simplification allows the application to avoid the task of assigning a unique file location for each simulation and the obligation of managing these historic results over time; however, the ability to retrieve previous simulations may be important for some cases.

3.3 **Enhanced PHP and Python Strategy**

Two changes to the system design allow many useful features to be added to the well permitting application. First, the geoprocessing tool is merged into the Python script. The second change involves a results file that the geoprocessing tool writes and the web application reads. These modifications demand implementation considerations discussed in Section 3.3.1 and allow the addition of specialized features that are outlined in Section 3.3.2.

3.3.1 Implementation with System Components

The two modifications to the system design are incorporated into Figure 3-3. Only the Python geoprocessing tool and the web application require significant changes.

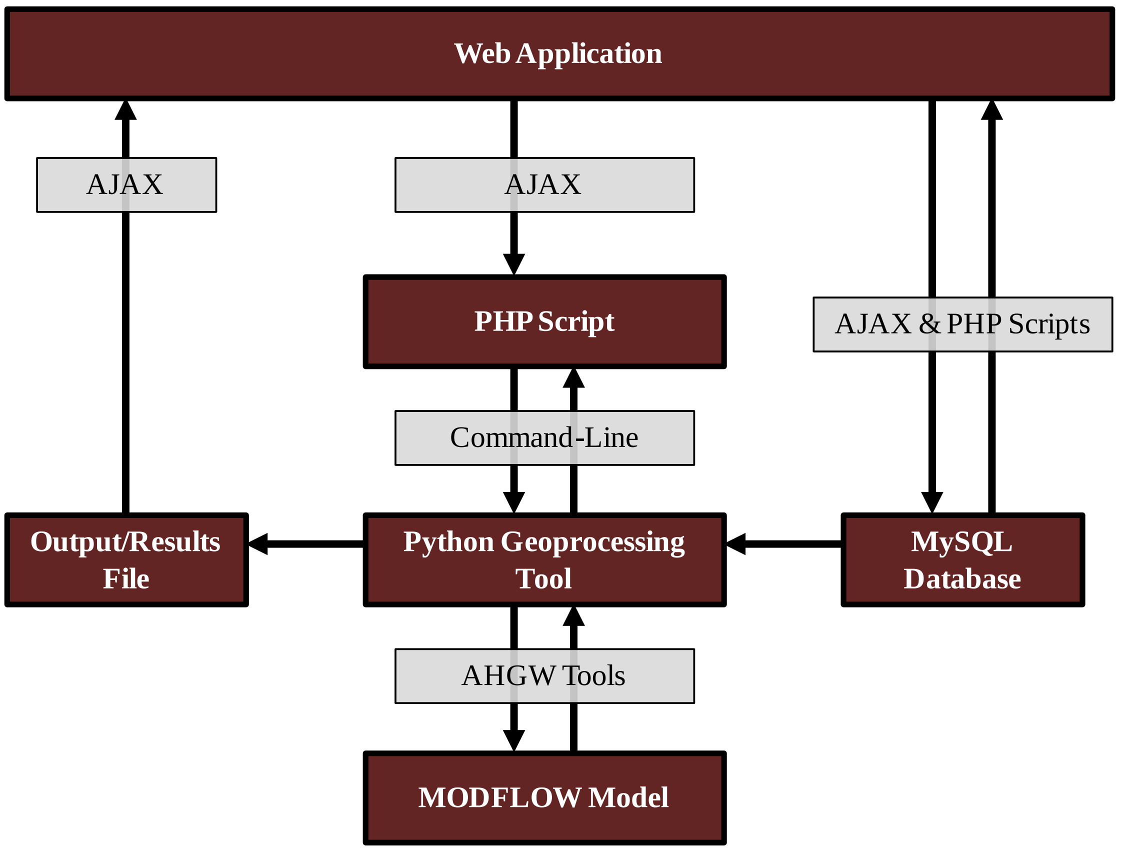


Figure 3-3. Data Flow Diagram for the Enhanced PHP/Python Method.

3.3.1.1 Python Script as Geoprocessing Tool

As discussed in Section 2.4.1, custom ArcGIS geoprocessing tools can be built as ModelBuilder workflows or as Python scripts, and Python scripts can either be designed as stand-alone tools that run from the command line or as ArcGIS-encapsulated tools that can only be executed using software like ArcMap or ArcCatalog. Because the geoprocessing tool must be autonomously initiated from a PHP script, the Python geoprocessing tool must be designed as a stand-alone tool.

A stand-alone geoprocessing script tool has much more flexibility than a ModelBuilder workflow. It can more easily incorporate loops, if statements, and other logic controls into the workflow. The capabilities of diverse module extensions can be exploited for highly customized functionality.

One of the first tasks the script must complete is retrieving the input arguments that define the Application ID to be analyzed and output options like contour interval. The *arcpy.getParameter()* functions are not applicable to stand-alone scripts; rather, the *sys* module is used in the script to retrieve the input arguments. The basic syntax for calling a Python script from the command line requires the Python file name followed by script arguments:

```
WellPermittingTool.py '1001' '1'
```

The string '1001' is stored as `sys.argv[1]` in the Python script, and the string '1' is stored as `sys.argv[2]`. `sys.argv[0]` stores the Python script name, either a full path or just the value entered in the command line, depending on the operating system. The Windows Server 2008 operating system, used for the case study implementation, returns the full path of the script. It is useful to be able to access the full path to the directory containing the Python script so relative references can be used for the data sources.

After retrieving the script arguments, the script creates the results file. The results file must be generated or cleared soon after the script initiates to ensure it exists before the web application starts repeatedly reading it. Preliminary tasks that take several seconds to complete, such as importing the *arcpy* module and importing custom toolboxes, are addressed after the initialization of the results file.

For custom tools – such as the AHGW tools – to be usable in a stand-alone script, the encapsulating toolbox must first be imported by the script, loading into memory the information required to use the tools. For faster import execution, a toolbox with only the required AHGW tools should be imported rather than the entire AHGW toolbox. Any other custom geoprocessing tools, such as the script used to query candidate well attributes using an Application ID, should also be added to this toolbox and imported. If ArcGIS extensions such as Spatial Analyst or 3D Analyst are required, they must be explicitly enabled. These issues are addressed automatically when a ModelBuilder workflow is used.

After the script addresses each of the preliminary tasks, it begins executing the well permitting workflow. Rather than merely saving KMZ files and other output to pre-specified locations anticipated by the web interface, the enhanced PHP and Python method requires the geoprocessing tool to communicate with increased flexibility with the web interface. A geoprocessing results file is used as an intermediary for this purpose.

3.3.1.2 Geoprocessing Results File

The result file is a simple text file written by the stand-alone Python geoprocessing tool and is interpreted by the web application. As the geoprocessing tool is running, it periodically updates the results file with status messages, links to KMZ results, and additional information as it becomes available. After a user initiates the tool execution, the web application checks the results file every couple of seconds; if the file has a new status message, the web application displays it; likewise, if the file has one or more new KMZ results, the web application adds it to the display.

The result file must be written in a consistent format that can be parsed by applications seeking the tool output. Each line of the results file stores a unique result message, and each message consists of components separated by commas. The message type is always identified by the first component, and identifies one of the following message type classes: background information, geoprocessing status updates, KML results, a HTML results, or termination status.

The number and expected format of following components depend on the message type, as outlined in Table 3-2.

Table 3-2. Definition of Result File Messages Structure

Component 1	Component 2	Component 3	Component 4
INFO	Text describing the information		
STATUS	Text describing the status		
KML_RESULT	Label or layer name	HTML content	URL of KMZ file
HTML_RESULT	HTML content		
END	Text describing whether the tool was successful or failed		

The background information (INFO) message type is used to store information like start-time and the input parameters, data that should be stored but that the user doesn't necessarily need displayed. STATUS messages are added to the results file at certain points in the Python script to update the user on the progress of the tool. Like the background information message type, STATUS messages only have one component after the type identifier, simply a text string of the status message. KML_RESULT messages have several components. The web application requires the URL of the resultant KMZ file and the layer label. Supplied HTML content will display beneath the layer label, possibly showing a legend or any necessary layer descriptions. A HTML_RESULT message instructs the web application to add the supplied HTML content to the page, but without adding any KMZ features. This could be used to highlight important numeric or text results, possibly the area surrounded by a certain drawdown contour, or to provide links to download output files. Finally, if the geoprocessing tool is no longer running, the results file should conclude with an END message that tells the web application to quite checking for updates and describes whether the tool completed without errors.

3.3.1.3 Web Application Configuration

For the elementary system configuration, the web application only needs to access the URL of the PHP script and wait for the results. The enhanced configuration requires more modifications to the web application. After accessing the URL of the PHP script, the web application turns its attention to the results file rather than waiting for results from the PHP page. The application must be programmed to parse the results file and take the appropriate action for each message.

When the web application encounters a status message from the results file, it updates the geoprocessing status of the results region of the web application.

When the application encounters a KML_RESULT message, it downloads the KMZ file and adds it to the map as before. A checkbox controlling the visibility of the new results layer and a customized Dojo title pane object are created on the fly and placed next to each other in the results area. The title pane initially only displays the layer label, but expands when it is clicked to display the HTML content passed from the results file. Results are added to the interface in the same order they are listed in the results file, somewhat limiting the layout control.

When the application encounters an HTML_RESULT message, the supplied HTML is similarly added to the results table of contents area of the web application. This is used to show text or links that do not correspond to a KMZ file. The enhanced Python script facilitates great flexibility and customization of results presentation; the HTML_RESULT message is provided to match this flexibility. A checkbox is not needed nor created, distinguishing the HTML result entry from KML results.

When the final message is interpreted and no END message has been encountered, the web application arranges to check for updates in a few seconds. On the other hand, if an END message is found, a final status message will be added and the results file will not be checked again.

This system configuration is conducive to adding stored simulation results to the web interface map since the same function can be used to interpret a stored results file. Of course, the URL references in the results file must be maintained.

3.3.2 Possible Python Script Enhancements

Some limitations and difficulties inherent in the strict ArcGIS ModelBuilder approach to custom automated geoprocessing may be overcome with Python scripting. This section will describe several enhancements to the geoprocessing tool possible with a stand-alone Python geoprocessing script.

3.3.2.1 Generate PDF Output

While the Google Earth-based map rendering provides a simple and intuitive interface for users of the web application, a more formal and permanent form of documentation is generally needed to communicate the results of the simulation. To address this need, a geoprocessing tool can be configured to produce print-quality PDF report with maps of the results. The *arcpy.mapping* module for ArcGIS 10 allows Python script to automate the processes required to add a formatted layer file to a pre-created template map document, adjust the positioning and text of map elements, and save the resulting document as a PDF file. If the template map document includes a legend, layers that are added programmatically to the map will also be added to the legend.

The symbology best suited for generating KMZ files for Google Earth differs from the ideal symbology for PDF map prints. Lines and points converted to a KMZ file and viewed on the web application appear thinner and smaller compared to the lines and points using the same symbology exported to a PDF. Other cartographic considerations may also require specialized styling be applied to an output layer before it is added to a PDF map. This may easily be accomplished within the geoprocessing Python script.

The *arcpy.mapping* module has several additional capabilities worth mentioning. Multiple PDF document may be appended together, allowing a multi-page report to be generated, perhaps displaying drawdown for each MODFLOW layer on separate pages. Map document elements may be repositioned on the page, and text can be modified. Maps may be exported to many different image formats, not only PDF. A complete documentation of the *arcpy.mapping* module is available online [16]. It should be noted that some features, such as configuring feature labels, lack the same degree of control that could be obtained manually with ArcMap.

3.3.2.2 Create Legends for KMZ Output

It may be desirable to display a graphical legend indicating the symbology associated with geographical result features. These legend graphics could be placed in collapsible title pane elements within the table of contents in the web interface. Unfortunately, the KMZ symbology color assignments for layers like drawdown and change in spring flow are not static, but rather were made to depend on the results for the particular simulation. Further, the *Layer to KML* geoprocessing tool does not generate associated legends.

The same *arcpy.mapping* Python module that facilitates PDF map output can be used to generate specialize legend images for each output KMZ file. The approach is to make a simple

ArcGIS map document with a customized page layout that exports only a legend object. After the geoprocessing tool creates the output feature, converts it to a layer, and applies symbology, the layer may be added to the legend map document; this will add the layer symbology to the legend object, and then the map can be exported to an image format, perhaps JPEG or PNG, rather than PDF. The URL of this image is then referenced by the HTML included in the results file for the KMZ results.

3.3.2.3 Embedded HTML Output

PDF results obviously cannot be displayed on the Google Earth map interface. As previously described, the results file allows any HTML to be passed from the geoprocessing tool into the web application. Although embedded HTML output capabilities were only utilized by the case study web application to provide the web interface user a link to PDF results, there are other compelling HTML outputs that could be generated by the geoprocessing tool. An application could easily be designed to print out non-geographic results as simple text strings or with HTML tables or images.

3.3.2.4 Display Results Immediately

Because the web application reads the results file several times while the tool is still running, it is possible to add simulation results as soon as they become ready. This is unlike both the ArcGIS Service method and the elementary PHP/Python method where all the results were returned at the same time; it effectively reduces the time a user must wait before beginning result interpretation.

3.3.2.5 User-Selected Output

Another possible feature that may be added to a geoprocessing Python script is allowing the user to select which output to compute and which output is not needed. The extended time required for the tool to generate results makes this an attractive capability. It is easily implemented with Python, only requiring additional input arguments and the inclusion of simple if statements surrounding the part of the workflow to be made optional.

3.3.3 Summary

The primary difference between the enhanced and basic PHP and Python strategies is the results retrieval method. The enhanced results retrieval method provides several features that would be unattainable with the basic methods.

3.3.3.1 Advantages

The enhanced PHP and Python service technique vastly improves the capabilities of the well permitting tool. The result file provides a direct line of communication from the geoprocessing tool to the web application. This allows the web application to incorporate real-time status update, add result KMZ files with the URL defined in the message file, and add HTML content embedded in the results file. Having the Python script act as the geoprocessing tool allows much greater flexibility in what the tool generates and how it does it. For example, high quality PDF maps can be created, legends may be added, and the user may be enabled to select which output need to calculate.

The benefits from interoperability and independence from ArcGIS Server discussed in Section 3.2.2.1 also apply to this service strategy.

3.3.3.2 Limitations

Compared to the basic PHP and Python strategy, the primary drawback of the enhanced technique is maintainability. While Python code is more flexible and powerful than the ModelBuilder framework, it is perhaps more difficult to understand and to modify.

The enhancements do not replicate all the ArcGIS geoprocessing service capabilities. Both PHP and Python strategies do not natively support simultaneous users and are not provisioned to store simulation results for a prolonged period of time. Implementation of these features would require further customization.

4 CASE STUDY

A case study based on a recent permit application submitted the City of Saratoga Springs will be presented. This demonstration will illustrate how a server-based automated well permitting system could be leveraged in water management practice. The computational time required by the permitting tool will also be discussed.

4.1 Introduction to the Example Problem

In December of 2009 the City of Saratoga Springs submitted an “Application for Permanent Change of Water,” desiring to transfer the rights to divert 450 acre-feet of water from four agricultural wells southeast of Utah Lake to eleven wells northeast of Utah Lake and use the water for municipal purposes. In response to this proposed action, several parties submitted formal notices of protest, arguing that the proposed extraction would infringe on existing water rights by depleting the aquifer and decreasing flows in the Jordan River. [18]

UT-DWR eventually approved the application, with a stipulation. To prevent a greater reduction of Jordan River flows than could be attributed to the original pumping south of Utah Lake, approval was granted provided that the resulting depletion would not exceed the estimated historical depletion associated with the same water rights. The city was instructed to maintain records to prove that their depletion did not exceed a prescribed volume.

The case study will use the well permitting tool to investigate whether the results of an automated MODFLOW simulation support the decision made by UT-DWR; specifically, aquifer drawdown and decrease in spring flows near the proposed wells will be considered. The Decision Memorandum [18] cited USGS's Scientific Investigations Report 2008-5197 to counter the argument that the proposed modification would deplete the existing aquifer.

4.2 **Methods and Assumed Candidate Well Attributes**

Because of the great distance between the original wells southeast of Utah Lake and the proposed wells northeast of Utah Lake, only the proposed wells were analyzed. The original wells, in fact, lie in a distinct water rights policy area and are not within the region covered by the Northern Utah County MODFLOW model. Had these original well locations been closer to the proposed locations, the cessation of water diversion at these points may have been accounted for by adding a copy of the wells at these locations with mirrored (positive) pumping rates to simulate the removal of the wells from the model.

The automated well permitting tool requires information regarding candidate wells to be added to a database. The latitude, longitude, pumping rate, screen top elevation, and screen bottom elevation of each candidate well are required. UT-DWR uses the Public Land Survey system to represent point locations, and the unit of acre-feet to represent water rights holdings. Location values were converted to decimal degree latitudes and longitudes, and yearly water volume allotments were converted to cubic feet per day. The change application requests an allotment of 450 acre-feet per year be granted to the eleven proposed wells. Lacking details on how the allotment would be distributed, the analysis assumed a uniformly distributed pumping rate of 4,882 cubic feet per day. For this application, no screen elevation data was available. These elevations were estimated with the intention of locating the screens of each well in the

MODFLOW layer most likely to conduct the highest flow; layer 3 was identified as a relatively thick, conductive, and accessible layer. An alternative approach could be to execute multiple simulations with the wells placed at various depths.

Table 4-1 lists the candidate well records compiled for this single case study analysis. Each well has the same application identifier to indicate that all the records shown are part of the same water rights application. The latitude and longitude values are in decimal degrees, the flow is in cubic feet per day with the negative sign indicating extraction, and the screen elevations are in feet above sea level.

Table 4-1. MySQL Database Records Derived from Available Data

Well ID	Application ID	Latitude	Longitude	Flow (cfd)	Screen Top Elev. (ft)	Screen Botm. Elev. (ft)
23	36127	40.36255	-111.88699	-4882	4400	4000
24	36127	40.36516	-111.88421	-4882	4400	4000
25	36127	40.39560	-111.92313	-4882	4400	4000
26	36127	40.40061	-111.91242	-4882	4400	4000
27	36127	40.39311	-111.89718	-4882	4400	4000
28	36127	40.38820	-111.90139	-4882	4400	4000
29	36127	40.37765	-111.88800	-4882	4400	4000
30	36127	40.37030	-111.89188	-4882	4400	4000
31	36127	40.36522	-111.88950	-4882	4400	4000
32	36127	40.32007	-111.91004	-4882	4400	4000
33	36127	40.30969	-111.89203	-4882	4400	4000

4.3 **Tool Results**

After the required well parameters were entered into the database, the automated well permitting tool was used to estimate the drawdown and change in spring flows that would be expected after approval of Saratoga Springs' request. The proper Application ID and a drawdown contour interval of 0.2 feet were entered in the tool input forms as shown in Figure 4-1. After the Application ID was entered, the locations of candidate wells were displayed on the map for verification.

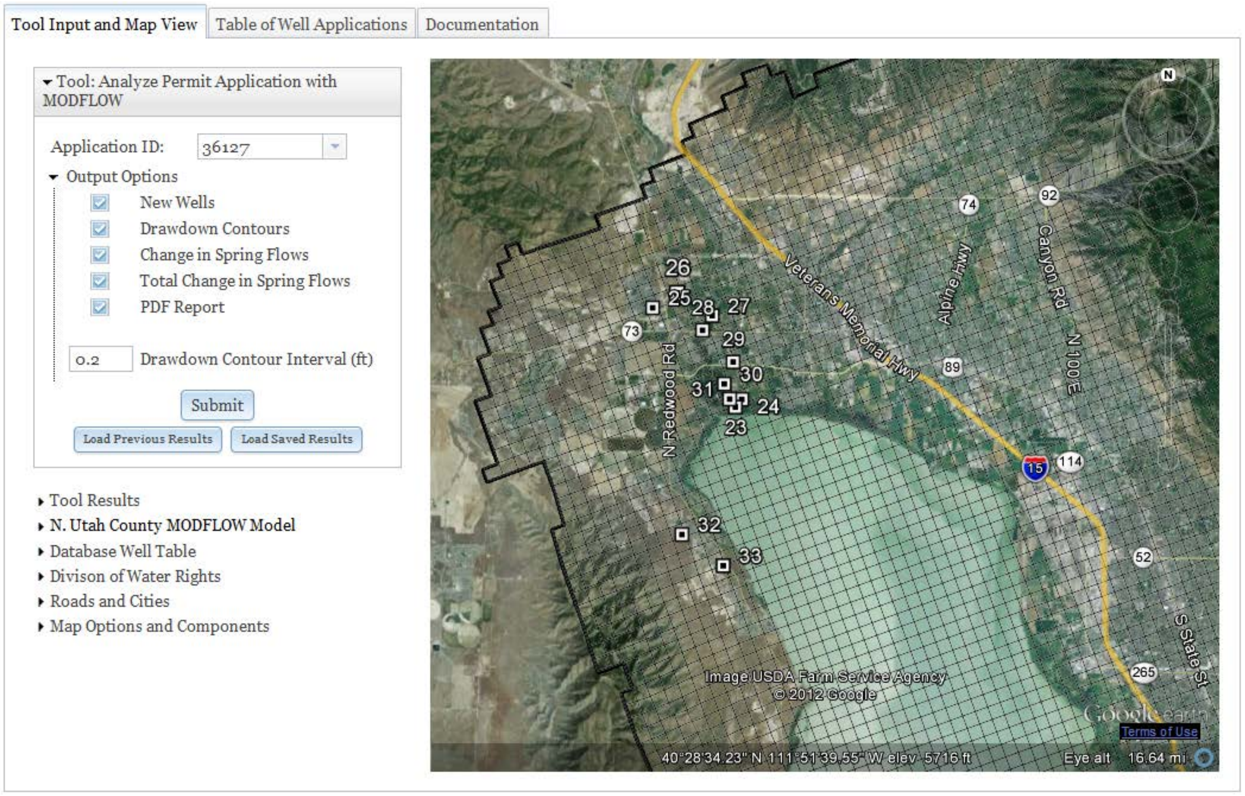


Figure 4-1. Tool Inputs and Locations of Candidate Wells

The submit button was clicked to execute the tool. After several minutes, the tool computed drawdown and spring flow and added result layers to the map. As expected, the greatest drawdown and spring flow decreases near the proposed wells were found in the results corresponding to MODFLOW layer 3, the layer where the wells were located due to the screen elevation assumptions. Figure 4-2 shows the drawdown contours results for layer 3. The largest computed drawdown contour was 1.6 feet.

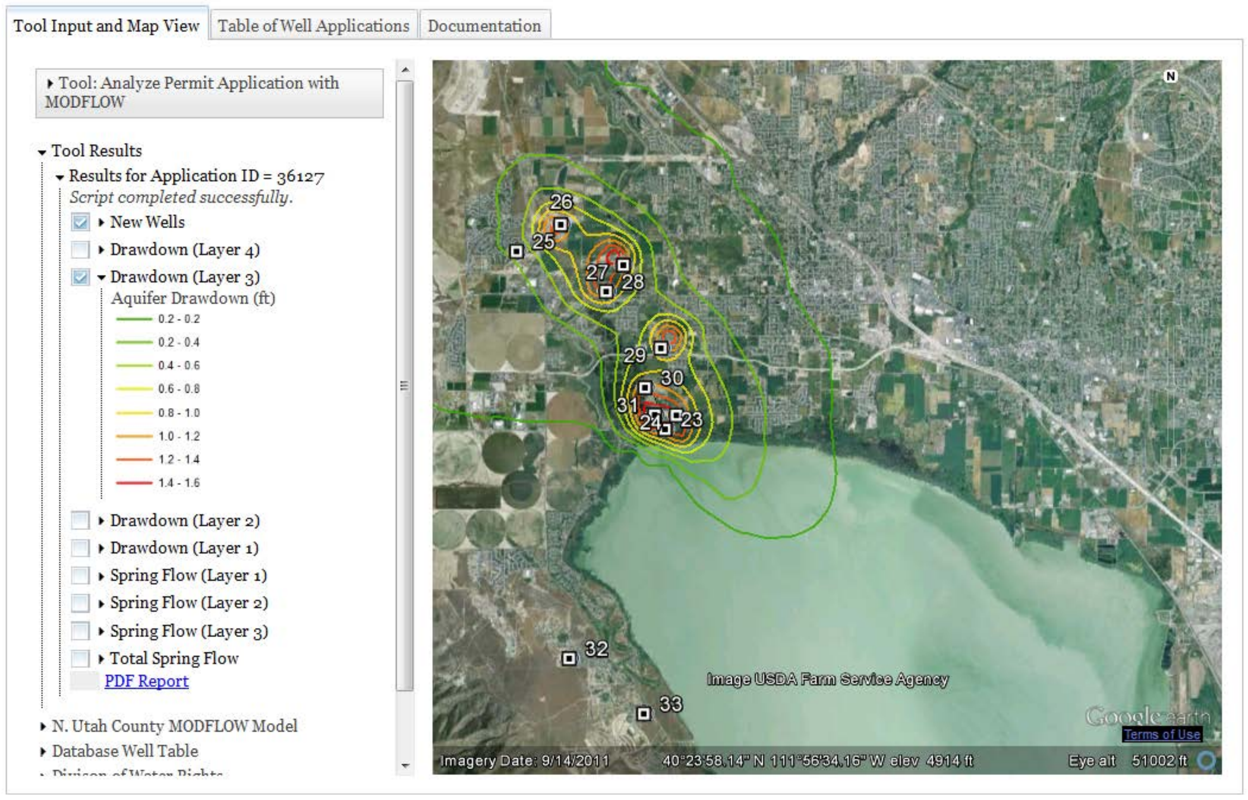


Figure 4-2. Results Generated by the Permitting Tool Showing Aquifer Drawdown in MODFLOW Layer 3 with Contours Increments of 0.2 Feet.

Other static map layers were used to verify the reasonableness of the drawdown results. Figure 4-3 shows the drawdown results along with the layer 3 hydraulic conductivity values used by the MODFLOW model. As expected, the anticipated drawdown is largest in locations where the new wells extract water from areas of low conductivity. As shown by Figure 4-3, the contour lines are not exactly centered over the candidate wells, but rather are centered over MODFLOW cell centers. This is a symptom of coarse MODFLOW grid resolution.

The UT-DWR maintains a particularly useful map layer that shows the locations of all authorized points of diversion. Figure 4-4 shows the expected drawdown contours overlaid with the locations of neighboring wells and other points of water diversion. Due to the vast number of wells in the neighborhood, this map shows only the northern-most drawdown area.

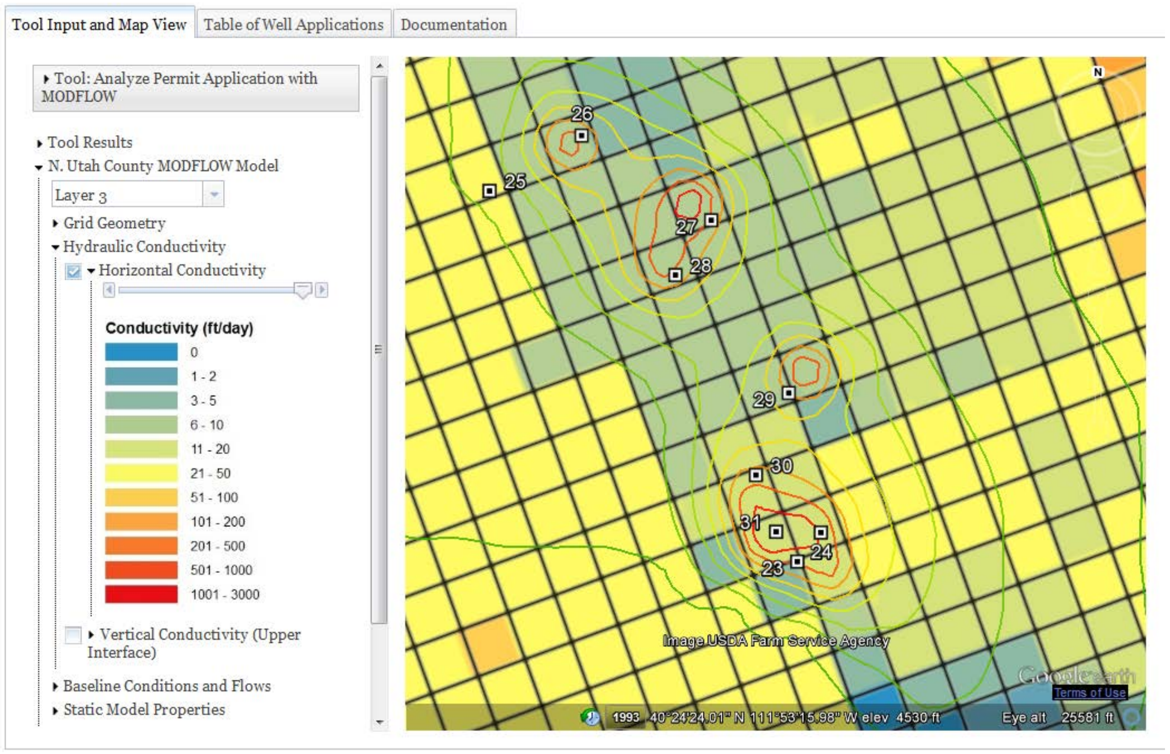


Figure 4-3. Drawdown Results Superimposed with Map of Horizontal Hydraulic Conductivity.

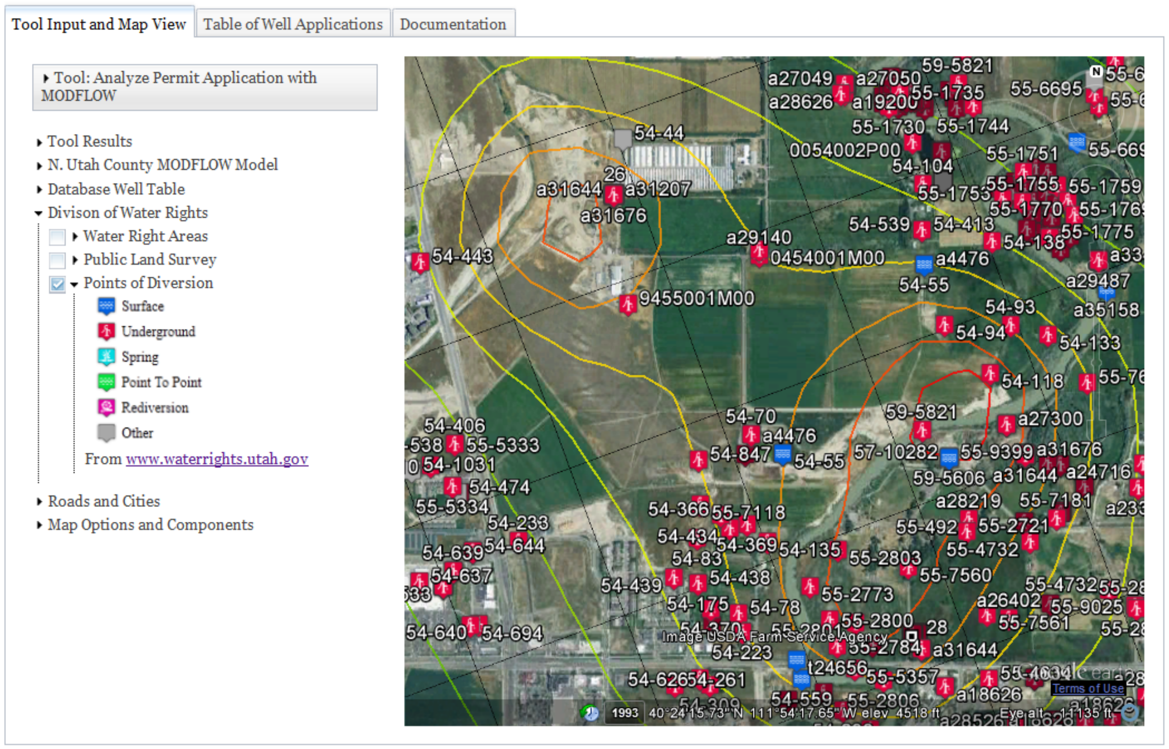


Figure 4-4. Drawdown Results Superimposed with Map of Existing Points of Diversion.

The decreases in spring flows for layer 3 are shown in Figure 4-5. The proximity of these springs to the Jordan River seems to validate the concern that the proposed action could impact this river, and to justify the depletion limitation stipulated within the Decision Memorandum.

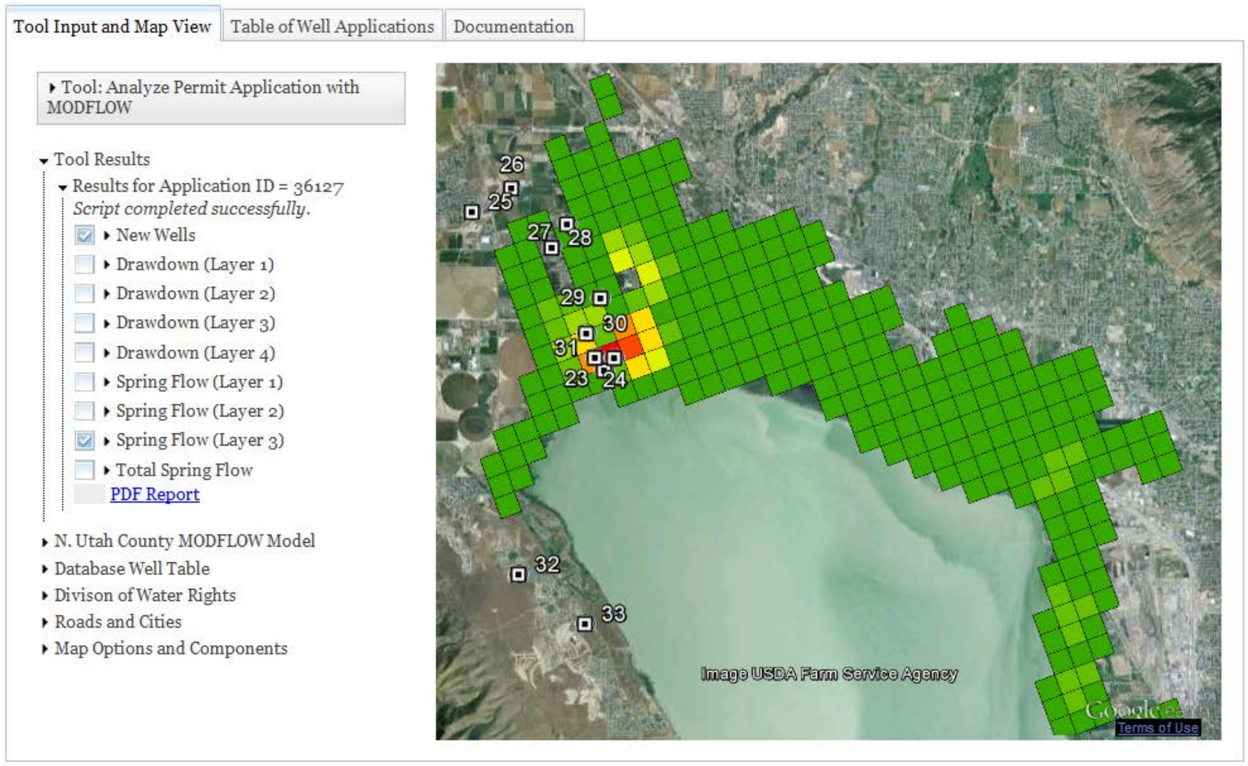


Figure 4-5. Results Generated by the Permitting Tool Showing Decrease in Spring Flows in MODFLOW Layer 3.

The springs are represented as MOFLOW drain elements. To facilitate verification of change in spring flow results, the web interface includes a drain conductance layer. Unusual results may then be investigated using the relationship between drawdown, change in flow, and conductance for the drain elements in question.

Suspicious results returned by the case study provide an example to demonstrate this utility. The change in spring flow for MODFLOW layer 1 reports substantial flow decreases in locations far from any candidate wells. Figure 4-6 shows these questionable results, marking three drain elements that will be investigated and verified.

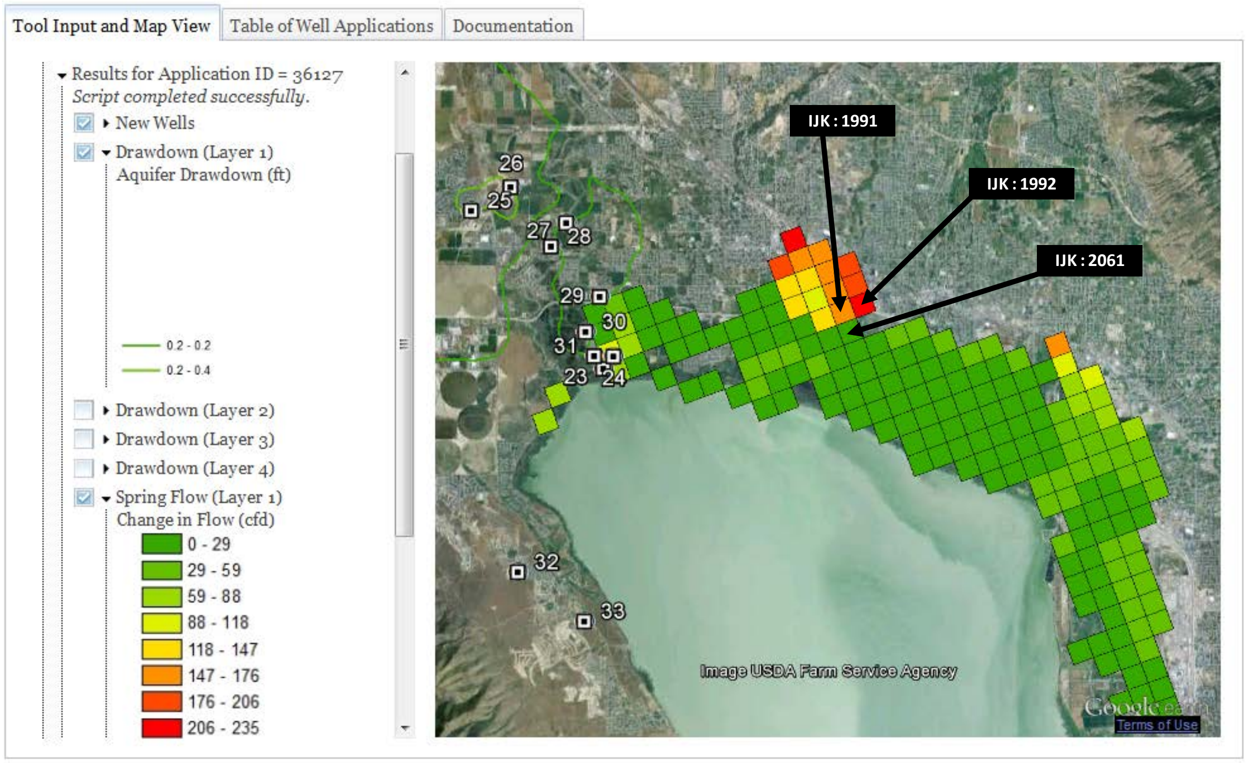


Figure 4-6. Change in Spring Flows and Drawdown Results for MODFLOW Layer 1 with Identification of Drain Elements to Be Verified.

For each of the three investigated drains, the change in flow was obtained from the results KMZ file, conductance was obtained from the general drain conductance KMZ file, and drawdown was back-calculated as the ratio of the change in flow and conductance. These values are listed in Table 4-2.

Table 4-2. Summary of Values Used to Verify Results for MODFLOW Layer 1 Change in Spring Flows

IJK	Conductance (ft/day)	Change in Flow (ft3/day)	Drawdown (ft)
1991	128,822	157.0	0.0012
1992	126,937	222.2	0.0018
2061	729	27.5	0.0377

The drawdown contour layer, with increments of 0.2 feet, establishes that the change in head at the drain elements must be less than 0.2 feet. A subsequent simulation was executed with a smaller contour interval and found that the back-calculated drawdown values match the simulated drawdown values. The unusual results are not due to a post-processing error, but rather to extremely large conductance values. The fidelity of the MODFLOW model to the actual system, however, may merit question. This exercise highlights the remaining importance of critical user evaluation of the results.

4.4 **Computational Time**

This discussion will be limited to two primary factors that impact the amount of time required to execute the well permitting tool: the design of the geoprocessing workflow – including what results to calculated, whether to create a PDF report document and whether to calculate results for every MODFLOW layer – and the processor of the machine running the tool.

To investigate the computational time required by the permitting tool, a PHP-Python system similar to the one described in Section 3.3 was used. The tool was made to generate KMZ and PDF results showing the locations of the input wells, drawdown contours for each

MODFLOW layer, change in spring flows for each layer, and total change in spring flows. The underlying Python script was slightly modified to print out the time required to complete each major geoprocessing tool. To facilitate testing with machines not configured as servers, the case study simulation was executed by the Python script independent of the PHP script and the web application. The time test was performed using three computers. The total required computational times that were obtained for each computer are listed in Table 4-3. The results show machines with multi-core processors reduce the required computational time.

Table 4-3. Comparison of Total Computational Time for Complete Case Study Simulations Using Different Computers

Computer	**Run Time (sec)**	**Operating System**	**Processor**	**RAM**
1	677	Windows Server 2008	Intel Xeon CPU X5650 @ 2.80 GHz, 2793 Mhz, 1 Core, 2 Logical Processors	8 GB
2	565	Windows 7	Intel Core 2 CPU 6600 @ 2.40 GHz, 2400 Mhz, 2 Cores, 2 Logical Processors	2 GB
3	467	Windows 7	Intel Xeon CPU X5650 @ 2.67 GHz, 2666 Mhz, 6 Cores, 12 Logical Processors	8 GB

Table 4-4 shows the time required by the three test computers to complete individual components of the geoprocessing workflow. Most of these components are geoprocessing tools, and several are repeated multiple times in the workflow in order to calculate results for each MODFLOW layer. Geoprocessing tools not listed required less than one second for each test.

Table 4-4. Required Computational Time of Individual Components of the Well Permitting Python Script for the Case Study by Three Different Computers.

Geoprocessing Tool/Command	**Type**	**Computer 1 Time (sec)**	**Computer 2 Time (sec)**	**Computer 3 Time (sec)**
Import Simulated Equivalents	AHGW	66.2	74.8	118.0
Run MODFLOW	AHGW	32.5	38.1	43.1
Create MODFLOW Well Records	AHGW	26.2	30.0	33.1
Create MODFLOW Features	AHGW	25.1	28.0	29.3
Export Package WEL	AHGW	21.7	26.9	29.8
Create PDF map page	Other	16.1	19.8	21.7
Import *arcpy* module	Other	7.0	6.3	9.3
Import Canidate Well Table	Custom	7.0	11.1	10.8
Import MODFLOW Output	AHGW	6.3	8.6	18.8
Join Field	ArcGIS	4.5	5.7	11.2
Import custom toolbox	Other	3.3	5.8	8.6
Natural Neighbor	ArcGIS	2.5	3.8	3.9
Copy Rows	ArcGIS	2.1	3.1	4.1
Copy Features	ArcGIS	1.9	3.1	3.6
Copy Rows	ArcGIS	1.6	1.9	3.2
Layer To KML	ArcGIS	1.5	2.0	3.9
Contour	ArcGIS	1.4	2.2	2.5
Dissolve	ArcGIS	1.2	2.0	2.3
Feature Class To Feature Class	ArcGIS	1.0	0.9	2.3
Calculate Field	ArcGIS	0.5	0.7	1.5

These exploratory results indicate which geoprocessing tools demand significant time resources. Most of these computational intensive tools are AHGW tools that are necessary to convert between MODFLOW input or result files. Optimized design of the well permitting tool can minimize the number of times these tools must be used. For example, in cases requiring change in spring flows for each MODFLOW layer, the *Import Simulated Equivalents* and *Create MODFLOW Features* tools should only be executed once rather than for each individual layer.

Also, the workflow may be modified for situations that don’t require results for both drawdown and spring flows, results for each MODFLOW layer, or a PDF summary of the results.

5 CONCLUSION

In closing, the objectives outlines in the initial chapter will be revisited for evaluation. Similarly, the project scope will be recalled and possible improvement and novel applications will be suggested.

5.1 **Evaluation of Objective Outcomes**

The objectives of the server-based well permitting decision support system, as defined in the opening chapter, called for a maintainable and changeable configuration, the capacity to processes multi-well applications, and a powerful but user-friendly web application.

Maintainability was achieved by dividing the system into distinct components. Component interdependencies were limited, to varying degrees for each method described in Chapter 3, in order to confine potential modifications to a particular component. The web applications for each of the methods, aside from the basic PHP and Python method, load any result returned by the geoprocessing tool so the geoprocessing tool may be modified without affecting the web interface. Unfortunately, prospective changes to the input arguments for the geoprocessing tool would require pervasive modifications, affecting the web interface, the geoprocessing service, and the geoprocessing tool. If the desired change would involve required well attributes stored in the MySQL database, the geoprocessing tool and web application would only require modification.

The maintainability objective was also the incentive for investigating both ArcGIS Server and PHP/Python service strategies. The appropriate implementation depends on the capabilities, experience, and preferences of the implementing organization. ArcGIS Server software comes with significant training and documentation resources, but also with a significant price tag. The Python and PHP custom service strategies permit significant customization of the geoprocessing tool, even beyond customary ArcGIS geoprocessing capabilities, but the programming requires more effort and technical knowledge.

For the system to be applicable to real scenarios, it was imperative for the system to be capable of analyzing applications involving multiple wells. This objective was achieved by purposefully designing the well permitting workflow and by incorporating a candidate well database table. This MySQL table was used to store requires inputs and deliver them to the geoprocessing tool when queried.

A system that purports to offer increased accessibility to modeling capabilities must have a usable interface. JavaScript extensions such as Dojo and the Google Earth plug-in were incorporated for enhanced usability and functionality. The pursuit of this objective may have come in conflict with the goal of keeping the system maintainable; web programming techniques and requirements are liable to change dramatically over short periods of time as internet browsers evolve, and complicated application are more likely to be affected than simple applications.

5.2 **Further Work**

Nearly every component of the described system could be further developed and adapted. Similar systems could be developed reaching beyond the scope of this work for a variety of useful purposes.

The custom service strategies that were presented did not solve the simultaneous user problem. With further work, a PHP and Python geoprocessing service system could be developed to allow one user access to the geoprocessing tool at a time, with a queue to handle any concurrent requests. Additionally, some implementations may benefit from previous results being stored for a managed amount of time, a feature the described PHP and Python system doesn’t natively accommodate. Similarly, not every potential ArcGIS Server feature was explored. For instance, ArcSDE geodatabases were not utilized. ArcSDE includes database versioning capabilities that might facilitate simultaneous, concurrent geoprocessing tool usage, and other enhanced resource publishing.

Further development and enhancement of the AHGW tools would encourage future implementations. The significant computational time required by some AHGW tools may reduce the attractiveness of the decision support system for some cases. These existing tools could be streamlined for improved performance. More tools could be developed to provide additional capabilities. The ability to import MODPATH flow lines into a workflow, for example, could potentially be very useful to automated analyses.

This work exclusively utilized steady state MODFLOW simulations to analyze the long-term effects of new wells. Although well simulations are important for water management, MODFLOW models involve many other aquifer features and attributes that may be varied to simulate diverse scenarios. The same methods presented for serving automated well permitting simulations could be transferred to other groundwater modeling purposes. Applications could be developed, for example, to allow modifications to stream features, constant head features, or recharge. Such developments would primarily be implemented within the geoprocessing tool.

Further work could refine the web interface for increased usability, maintainability, and simplicity. The incorporation of sophisticated features such as custom widgets, a MySQL database for candidate wells, and the Google Earth map come at the expense of simplicity. The necessity of these components, and perhaps their maintenance costs, can be evaluated over time. While the ideal configuration will vary, future work could further explore the diverse and ever-changing design options for web maps.

Future development should fulfill the needs of water management decision makers. While the server-based well permitting decision support system pattern is a first step towards accessible groundwater modeling, more work remains.

REFERENCES

[1] Horsburgh, J.S., D.G. Tarboton, D.R. Maidment, and I. Zaslavsky, 2008. A relational model for environmental and water resources data. Water Resources Research 44:W05406.

[2] Horsburgh, J.S., D.G. Tarboton, M. Piasecki, D.R. Maidment, I. Zaslavsky, D. Valentine, and T. Whitenack, 2009. An integrated system for publishing environmental observations data. Environmental Modelling & Software 24:879.

[3] Castronova, A.M., and J.L. Goodall, 2010. A generic approach for developing process-level hydrologic modeling components. Environmental Modelling & Software 25:819-825.

[4] Molenaar, K.R., and A.D. Songer. 2001. Web-based decision support systems: Case study in project delivery. Journal of Computing in Civil Engineering 15:259-267.

[5] Dymond, R.L., B. Regmi, V.K. Lohani, and R. Dietz, 2004. Interdisciplinary web-enabled spatial decision support system for watershed management. Journal of Water Resources Planning & Management 130:290-300.

[6] Cheng, W.-C., M. Putti, D. R. Kendall, and W. W.-G. Yeh, 2011. A real-time groundwater management model using data assimilation. Water Resources Research 47:W06528.

[7] Oulidi, J., Hassane, R. Löwner, L. Benaabidate, and J. Wächter, 2009. HydrIS: An open source GIS decision support system for groundwater management (Morocco). Geo-Spatial Information Science 12:212-216.

[8] Tillman, F.D., S.A. Leake, M.E. Flynn, J.T. Cordova, and K.T. Schonauer, 2007. An online interactive map service for displaying ground-water conditions in Arizona: U.S. Geological Survey Open-File Report 2007-1436. http://pubs.usgs.gov/of/2007/1436/, accessed June 14, 2012.

[9] Jones, N.L., G. Strassberg, and A. Lemon, 2010. Automated Well Permitting via GIS Geoprocessing Tools. ASCE Conf. Proc. doi:10.1061/41114(371)74. World Environmental and Water Resources Congress 2010: Challenges of Change Proceedings of the World Environmental and Water Resources Congress 2010.

[10] Strassberg, G. 2005. A geographic data model for groundwater systems. Ph.D. diss., Department of Civil Engineering, The University of Texas at Austin.

[11] Cederberg, J.R., P.M. Gardner, and S.A. Thiros, 2009. Hydrology of Northern Utah Valley, Utah County, Utah, 1975-2005: U.S. Geological Survey Scientific Investigations Report 2008-5197, 114 p.

[12] Díaz, L., S. Costa, C. Granell, and M. Gould, 2007. Migrating geoprocessing routines to web services for water resource management applications. In M. Wachowicz & L. Bodum (Eds.), AGILE 2007 Proceedings. Aalborg: AGILE.

[13] Strassberg, G., 2011. *Arc hydro groundwater : GIS for hydrogeology*. Ed. Norman L. Jones and David R Maidment. Redlands, Calif.: ESRI Press.

[14] KML Support. Google. https://developers.google.com/kml/documentation/mapsSupport

[15] ArcGIS Server REST API. ESRI. http://help.arcgis.com/en/arcgisserver/10.0/apis/rest/

[16] ArcGIS Help. ESRI. http://resources.arcgis.com/content/web-based-help

[17] Dojo Toolkit. The Dojo Foundation. http://dojotoolkit.org/

[18] Utah Department of Natural Resources Division of Water Rights, 2010. “Order of the State Engineer: For Permanent Change Application Number 53-1686 (a36127)” http://waterrights.utah.gov/docImport/0530/05309227.pdf