TECHNICAL PROGRESS REPORT

Remediation and Treatment Technology Development and Support for DOE Oak Ridge Office: Python Scripts and ModelBuilder Process Workflow Diagrams for Geodatabase Development

Date submitted:

February 1, 2013

Principal Investigators:

Leonel E. Lagos, Ph.D., PMP®
David Roelant, Ph.D.

Florida International University Collaborators:

Georgio Tachiev, PhD, PE, Project Manager
Angelique Lawrence, MS, GISP

Submitted to:

U.S. Department of Energy
Office of Environmental Management
Under Grant # DE-EM0000598

Applied Research Center
FLORIDA INTERNATIONAL UNIVERSITY

Executive Summary

During FY11 (FIU Year 2), researchers at the Applied Research Center (ARC) at Florida International University (FIU) developed a geodatabase to support the hydrological work being performed at Oak Ridge Reservation (ORR), which serves as a centralized data management system, making the large amounts of data generated from the simulations of contaminant fate and transport accessible to all users. The geodatabase facilitates storage, concurrent editing and import/export of model configuration and output data. The work for FY12 (FIU Year 3) extended the geodatabase capabilities and created models using ArcGIS ModelBuilder and Python scripting that automate the process of querying the existing EFPC geodatabase and the generation of maps. Investigation of easily downloadable free/open source geographic information systems (GIS) software for viewing and querying the hydrological modeling data and for generating maps, graphs and reports, is now being conducted to determine a simple way of sharing project derived data with other project stakeholders such as U.S. Department of Energy (DOE) personnel and ORR site contractors.

Table of Contents

List of Tables

List of Figures

Introduction

During FY11, FIU developed a geodatabase to support the hydrological modeling work performed by FIU, which involves the use of three integrated watershed models for the Y-12 NSC, White Oak Creek (WOC), and East Fork Poplar Creek (EFPC) to simulate the fate and transport of contaminants. More than a hundred simulations were completed for the purpose of calibrating the models, deriving model uncertainties, and for providing the analysis of remediation scenarios, resulting in gigabytes of simulation data. Therefore, there was a need for an advanced spatial data structure that would be used to address the management, processing, and analysis of spatial and temporal numerical modeling data derived from multiple sources, and to produce hydrogeological maps for visualization.

The ORR Geodatabase is a multiuser relational database management system (RDBMS) built upon a Microsoft SQL Server platform developed using Environmental Systems Research Institute (ESRI) ArcSDE technology. The system was deployed on an advanced Windows server with the latest technology and hardware and provides a user interface which facilitates data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. The ORR Geodatabase is based on the ArcHydro and ArcGIS Base Map data models. The ArcHydro data model contains a set of accompanying tools designed to support water resources applications within the ArcGIS environment. These data models were also used as templates as there were many input data types in common with the ORR Geodatabase. Modifications were then made for project specific input parameters. The geodatabase serves as a centralized data management system, making the mass amounts of data generated from the simulations of contaminant fate and transport accessible to all users and facilitates storage, concurrent editing and import/export of model configuration and output data.

Technical Approach

The objective of this task during FY12, is to extend the geodatabase capabilities by creating a GIS-based model using ArcGIS ModelBuilder as well as Python scripting that will automate repetitive processes such as: (1) querying the existing ORR Geodatabase for files used in hydrological model development and performance of numerical simulations; (2) performing statistical analyses based on specified algorithms; and (3) generating maps with the spatial distribution of computed and observed data.

The use of PYTHON scripts provides capabilities including:

- Use of ArcGIS python geoprocessing libraries to iterate over any selected feature and to provide immediate analysis, updates, and modifications of the geodatabase.
- Creation of customized functions to handle various aspects of the analysis, creating more classes to implement object oriented capability and to promote code reuse within the application.
- Preprocessing of input shapefiles (sewer lines, nodes, links and any infrastructure) and creation of new shapefiles on the fly. This includes analysis of the attributes of the input files and creating of a new set of shapefiles using selected attributes.
- Iteration through a list of infrastructure elements (links, lines, nodes) selected by a polygon shapefile and for each selected component analysis of a number of attributes in the shapefiles, including total length of lines that will remain, are missing, undersized or to be abandoned, which provides analytic capabilities to derive information about each specific file.
- Automatic addition of multiple attributes to point, polygon and polyline shapefiles, including attributes that will be used for project tracking.
- Additional post-processing of relevant shapefiles, including selection, printing, formatting with a toolbox specifically designed to minimize the manual set up and efforts.

- Automatic zooming, mapping and exporting of selected spatial features to excel, pdf and CAD formats as needed.
- Creation of a special toolbox with required tools to provide GUI access to data analysis, creation of new shapefiles based on input, printing and merging information without requiring the user to spend time on entering data separately.
- Error checking and analysis of existing shapefiles and preparation for use in XPSWWM models (particularly nodes and lines).
- Creation of reports on information contained in map documents such as data frame coordinate system, layer data sources, layers with broken data sources, or layout element positioning; update, repair, or replacement of layer data sources in a map document or layer file; and update of layer symbology without having to physically open map documents.
- Creation of geographic data in batch using map export commands, such as a series of GeoTIFF images driven by a list of features in a data frame. Automation of the creation and management of map services to be published with ArcGIS Server.
- Building a variety of PDF map books, including a thematic or temporal map book with title page, multiple map pages, and any number of additional pages with supporting content such as tabular reports and contact lists. A reference map book based on Data Driven Pages which allows rapid processing and export of a series of layout pages from the map document (and selected ATLAS sheets).

Development of customized Python scripts for the purposes of this task require additional programming of built-in automated geoprocessing tools to call or retrieve data from the ORR geodatabase. The toolbox being developed will allow the ArcGIS program to iterate over a several GIS files for pre- or post-processing of data to be used in or that are derived from hydrological modeling. ArcGIS ModelBuilder will then be used to document the processes and workflows employed.

Figure 1 below, shows the overall ORR Geodatabase schema generated using the ArcGIS "Geodatabase Diagrammer" utility for ArcGIS 10. More details are visible in Figures 2-4. These images basically depict the data structure of the ORR geodatabase and the feature datasets and feature classes that are used or generated during hydrological model development as specified in Table 1.

Table 1. Model Configuration Files Stored in the ORR Geodatabase

Spatial Data	**Characteristics Represented**
Admin_Features	EFPC, WOC, Y-12 & OSY Model domains (polygons)
	ORR Boundary
Admin_GRIDs	Model domains (GRIDs)
Conductivity_GRIDs	Hydraulic conductivity GRIDs
Contaminant_Conc_Features	Monitoring points (has associated timeseries attribute data)
	Plume Contours
Contaminant_Conc_GRIDs	Interpolated contaminant plumes (GRIDs)
DEMs	Clinch River, EFPC & WOC Watershed DEMs
Digital_Orthophotos	ORR DOQs (.bmp)
Drainage_GRIDs	Drainage Time Constant, Drainage Codes, Detention Storage (GRIDs)
GW_Features	Groundwater level contours
GW_GRIDs	Groundwater level GRIDs
Hydro Features	Watersheds, subwatersheds, catchments, hydroareas (lakes/ponds) (polygons)
	Floodplain polygons
	Hydrography, Hydrodrainage, hydrostructures (polylines)
Impervious_GRIDs	Paved runoff coefficient (GRID)
Landcover_Landuse_Features	Landuse/Landcover polygons
Landcover_Landuse_GRIDs	Vegetation grid codes
Manning	Manning's coefficients (GRIDs)
Monitoring_Stations	USGS SW monitoring stations, outfalls, GW monitoring wells
Network_Features	Rivers, streams, reaches, cross sections, diversion ditch, utilities (polylines)
	Nodes (points)
Physical_Features	Buildings, obscured areas, natural outlines, man-made outlines (polygons)
	Margins, man-made structures (polylines)
Soils	Geology, soils (polygons)
Topo_Features	Elevation contours
Transport_Features	Roads, railroads, transportation structures (polylines)

Temporal Data	**Characteristics Represented**
Monthly_RF_TS	Monthly rainfall timeseries
Flow_Aug_TS	Flow augmentation timeseries
DHI_Timeseries	Flow rate/discharge timeseries

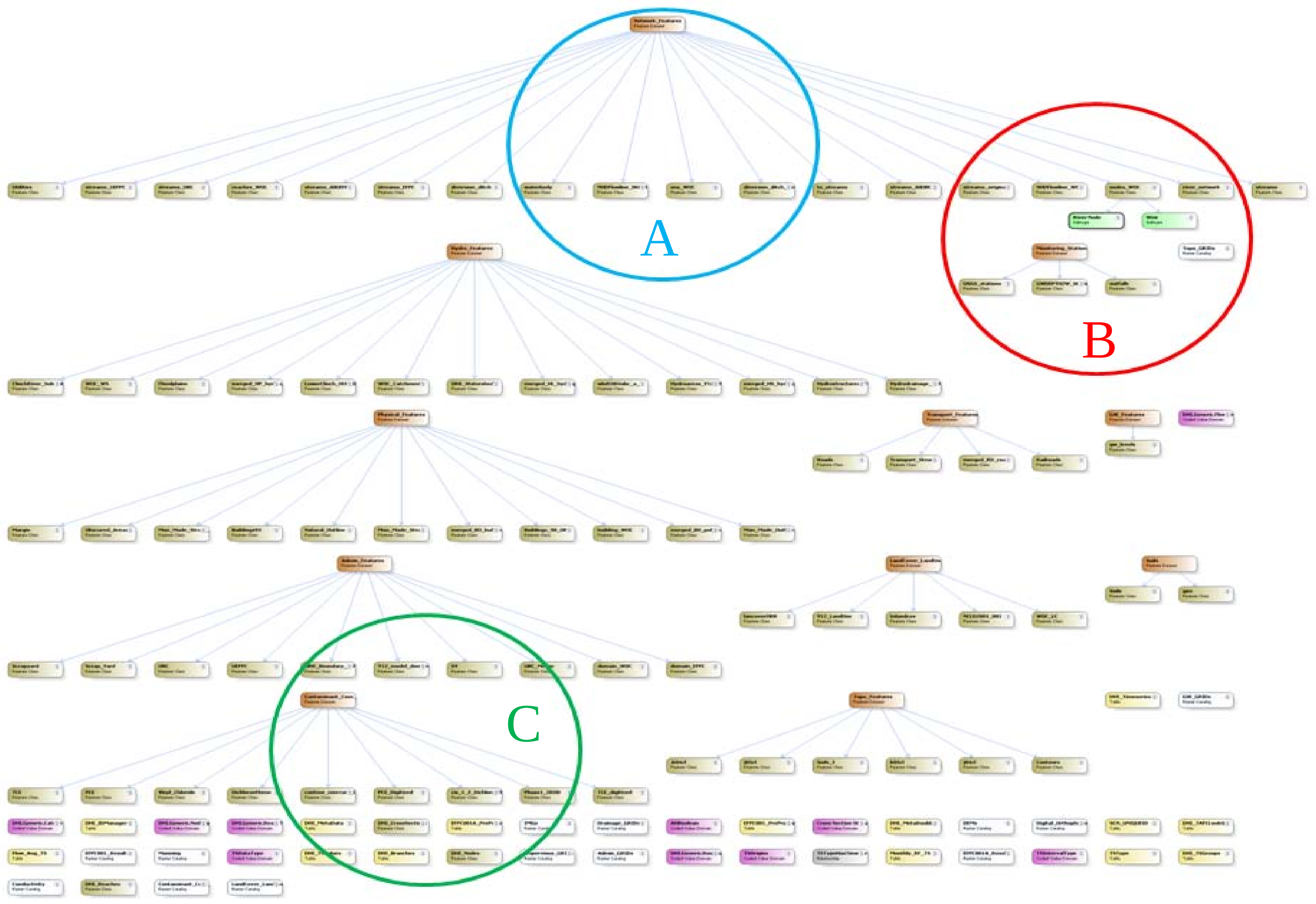

Figure 1. The ORR Geodatabase schema generated using the ArcGIS “Geodatabase Diagrammer” utility for ArcGIS 10.

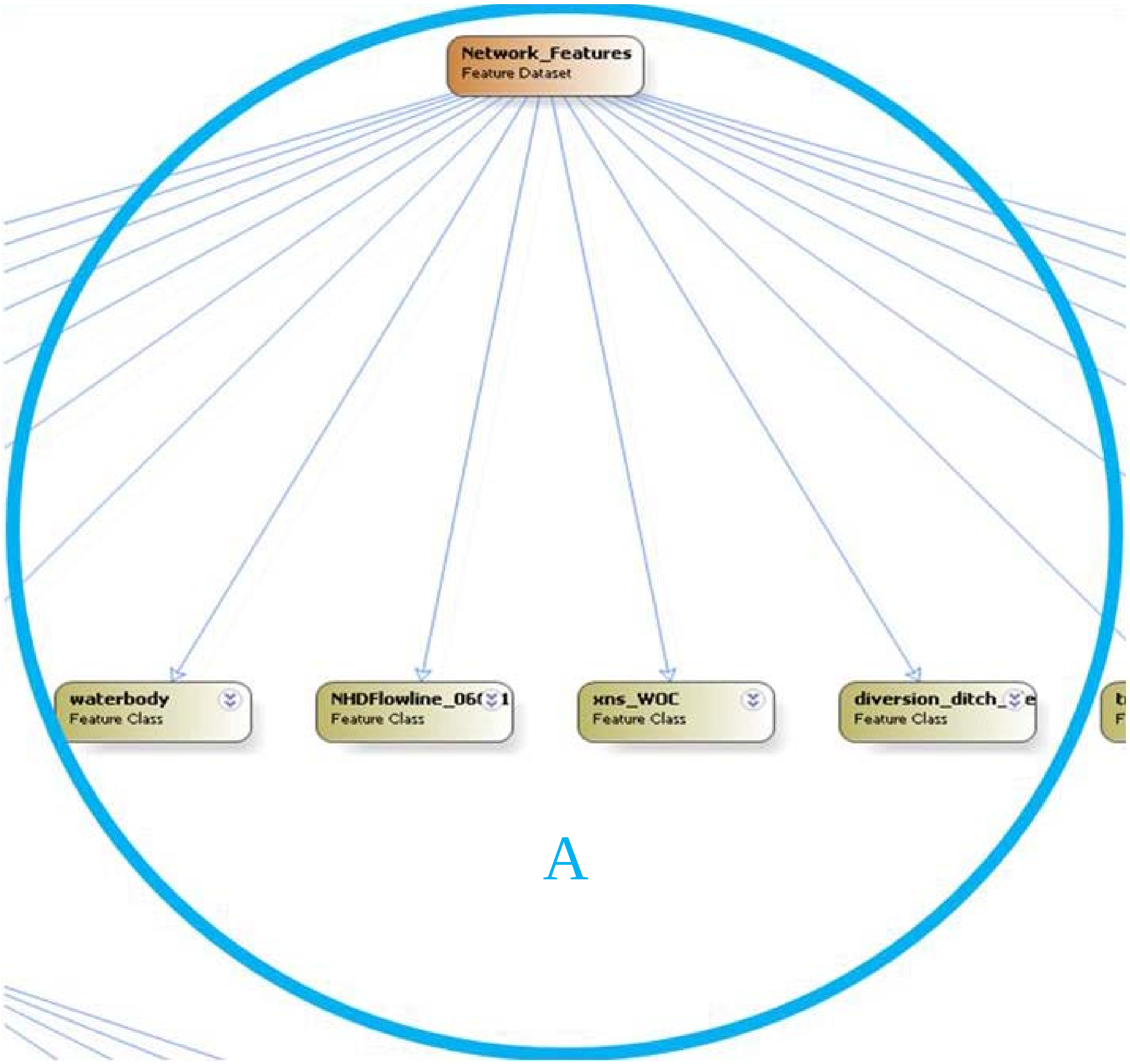

Figure 2. Closer view of circle A in the schema diagram shown in Figure 1 above.

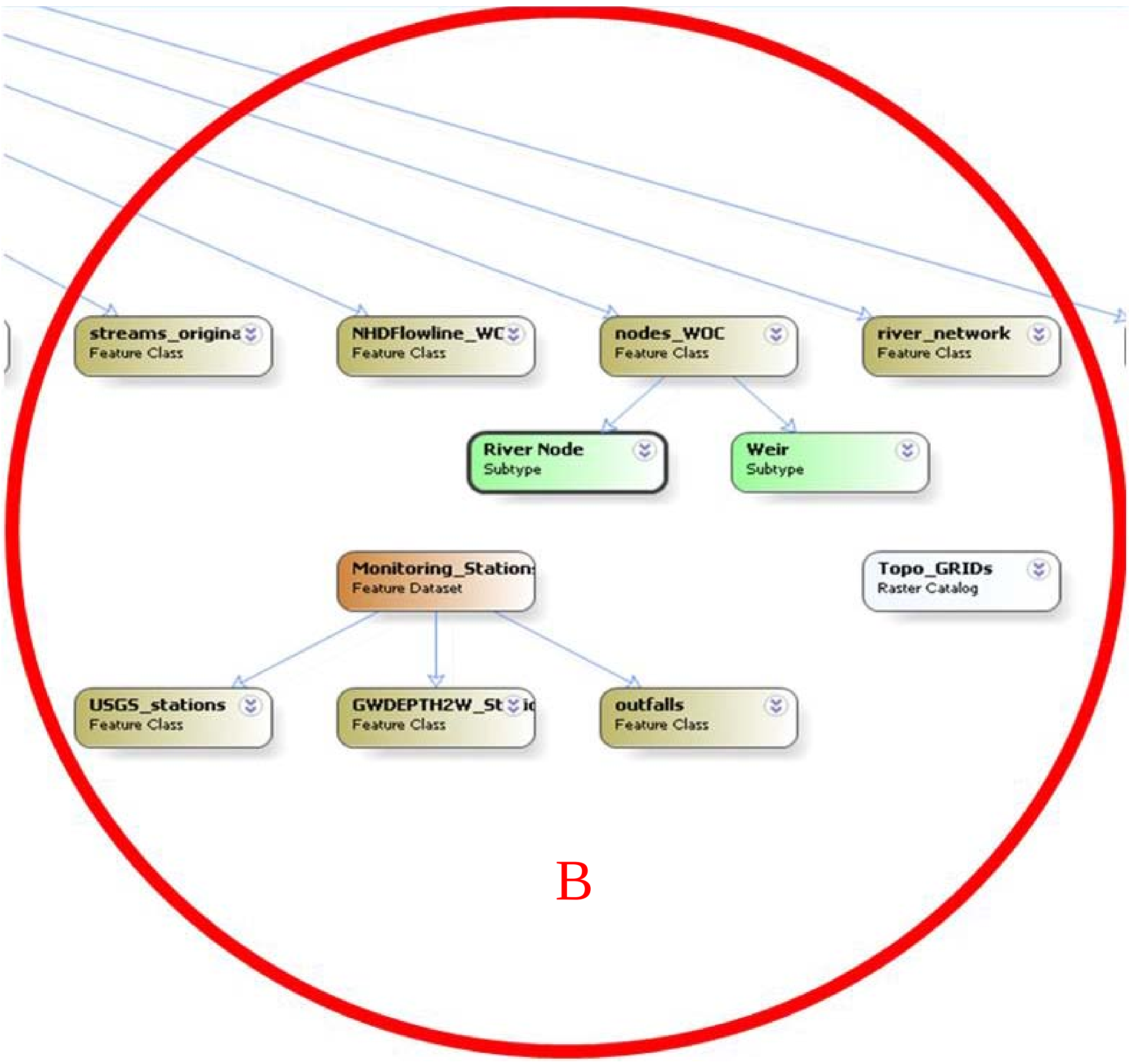

Figure 3. Closer view of circle B in the schema diagram shown in Figure 1 above.

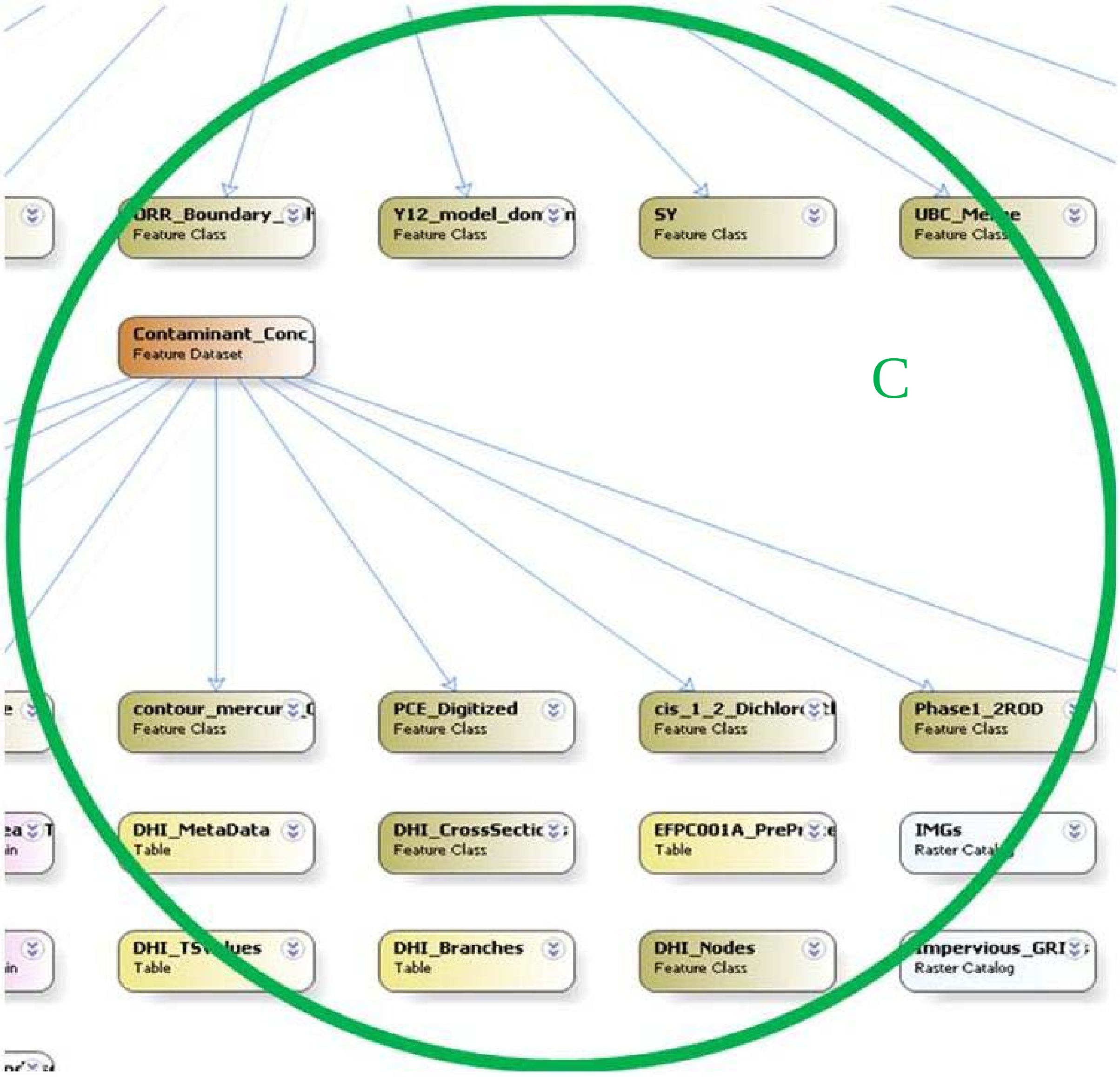

Figure 4. Closer view of circle C in the schema diagram shown in Figure 1 above.

The essence of this task is to provide modelers with ease of access to model data and pre-processing capability for files, like those featured in Figures 1-4, to be used as input into the MIKE SHE/11 models. Post-processing of model results is also possible for analysis, use in reports and map production. Simple tasks such as retrieving data from the geodatabase; pre-processing the data; exporting for use in hydrological model development; subsequent import and post-processing of model data; data analysis; and production of graphs, maps and reports are repetitive but necessary. The use of Python scripts and ArcGIS ModelBuilder assists in automating these tasks which saves time and can facilitate batch processing of this data. One of the objectives of this task, therefore, was to also develop a reusable GIS tool, which can iterate over each set of the spatial input data parameters perform geoprocessing actions and calculate statistical parameters.

A toolbox which combines built-in ArcGIS geoprocessing tools coupled with customized Python scripts is being developed specifically for use with the East Fork Poplar Creek (EFPC) model. These tools and scripts automate the query and retrieval of timeseries data, including contaminant flow and transport parameters (e.g. mercury concentration, surface water and groundwater flow, discharge, groundwater level, etc.), from the existing ORR geodatabase. The ArcGIS data model iterates through selected features and exports the results in tabular format. Existing Python scripts can also be incorporated for statistical analysis of the exported data. Once a feature (e.g. GW well or outfall) has been selected, a field attribute such as station name can be used to extract and export all the data for that station in MS Excel or text format. Geoprocessing such as interpolation of the extracted values and generation of raster images for each day in ESRI GRID and TIFF formats can also be automated. ModelBuilder tools and Python scripts are also available in the toolbox to enable the export of maps from an ArcMap document within a specified data frame in PDF format. In addition, ArcGIS ModelBuilder can be used to generate model workflow diagrams which are a great way of documenting and visually representing the geoprocessing tools and scripts being incorporated into the data model as development progresses.

The GIS tool enables analysis of spatial and temporal monitoring data at ORR. It is being developed as a set of GIS toolboxes that use a set of input GIS files. The tool is calibrated to the project's location (i.e. the EFPC model domain) and is scalable and reusable.

The GIS toolbox being developed in this project provides equivalent functionality and will have capabilities to:

i. Add GIS files to ArcMap and create layer files.

ii. Select features within a specified area (e.g. the study domain) and zoom to selected features.

iii. Clip/extract selected features and create new layer file of selected subset.

iv. Export clipped feature in format to be used by MIKE SHE/11 model.

v. Export attributes of clipped feature in MS Excel or text format for statistical analysis and generation of graphs and reports.

vi. Export map extent in various formats (e.g. JPEG, TIFF or PDF) for development of reports.

vii. Interpolate timeseries data collected at various monitoring points, generate gridded surfaces, and finally create and export mapped results (as seen in image below).

Progress Summary

Work for FY12 has involved the creation of a model using ArcGIS ModelBuilder and Python scripting that automates the process of querying the geodatabase based on specific environmental parameters, performs analyses based on specified algorithms and generates maps with the spatial distribution of computed and observed data.

Work completed to date includes:

- Background research for development of a draft Project Technical Plan (PTP) for proposed FY12 work scope related to the use of ModelBuilder and Python scripts within the ArcGIS environment for automating data import/export and for conducting certain geoprocessing tasks.
- A preliminary literature review for the use of Python scripting to automate various geoprocessing tasks and the use of ArcGIS ModelBuilder to generate process flow diagrams. This is to support external query and retrieval of mercury and hydrological model data from the existing ORR geodatabase.
- Literature review of the use of ArcGIS ModelBuilder coupled with Python scripting to automate various geoprocessing tasks and generate process flow diagrams. This is to support external query and retrieval of mercury and hydrological model data from the existing ORR geodatabase. The following are some of the documents, presentations and technical workshops reviewed:

    1. ESRI International User Conference Technical Workshops:
        - “Model Builder Advanced Techniques,” Scott Murrary, July 2010.
        - “Working with Temporal Data in ArcGIS,” David Kaiser, Hardeep Bajwa, July 2010.
    2. ESRI Southeast Regional User Group (SERUG) Conference 2010 Technical Workshop:
        - “Intermediate ModelBuilder,” Kevin Armstrong.
    3. Wikihow: “Creating time-series raster mosaics in ArcGIS 10 for Eye on Earth.”

4. "Model Builder Lab," Geoinformatics, Spring 2008, Purdue University Library.
5. "Time-Series Contaminant Interpolation using ArcGIS and Spatial Analyst," Mark K. Petersen, ESRI User Conference Proceedings 2006, Paper 1326.

- Utilization of the aforementioned resources to develop and test a preliminary model using ArcGIS ModelBuilder coupled with Python scripts which:
  - Automates the retrieval of groundwater level daily timeseries well data derived from OREIS by date
  - Interpolates the extracted values, and
  - Generates raster images for each day in ESRI GRID and TIFF formats.
- Research to assist in development of Python scripts to enable the raster images produced to be stored in a raster catalog archived by date to facilitate visualization and animation of the temporal changes in groundwater levels for the specified study domain over a given timeframe. To date the scripts developed have enabled storage of the raster images produced in a raster catalog, however, further development is necessary for automated archival of these images by date.
- Development of a separate model using ModelBuilder and Python scripting to enable the export of maps from an ArcMap document within a specified data frame in PDF format. Refinement of this model is now being conducted.
- Research and testing of various Python scripts for automated archival of interpolated raster images by date in a raster catalog.
- Integration of Python scripts from an already existing model into the simplified model created to export maps generated from files within the ORR geodatabase in PDF format. This is in an attempt to further refine the model to enable users to export maps from an ArcMap document within a specified data frame.
- Development of a model using ArcGIS ModelBuilder which iterates through selected features and exports the results in tabular format. This can be utilized to extract model input and output data contained in the geodatabase such as groundwater level, discharge and mercury concentration. Once the feature (e.g.

GW well or outfall) has been selected, a field attribute such as station name is used to extract all the data for that station and export it as a .dbf table which can be opened using MS Excel.

- This model is currently being extended and refined to enable greater functionality by developing new or modifying and incorporating existing Python scripts for statistical analysis of the exported data. This will be especially useful for extracting and analyzing timeseries data used in the EFPC model. Once the model is completed, a ModelBuilder workflow diagram will be generated and documented.
- Further development of Python Scripts to extend and refine model by enabling data to be extracted from the ORR geodatabase and saved/exported as Excel or .txt files.
- Implementation of a toolbox in ArcGIS which uses the scripts to analyze water stages in rivers, flow rates, groundwater levels and water quality data from wells and rivers.
- Testing of the toolbox capabilities to extract and analyze timeseries data used in the EFPC model. Once the model is completed, a ModelBuilder workflow diagram will be generated and documented.

Future Work

Future work to be carried out to the end of FY12 includes research related to an extension of the geodatabase capabilities to facilitate online querying of the database using downloadable freeware and generation of maps, graphs and reports. This includes:

- A literature and Internet search of downloadable GIS freeware that can be used for querying the ORR geodatabase online.
- Development & testing of a methodology for querying ORR geodatabase via online GIS freeware.
- Compilation of all research, references, Python scripts and ModelBuilder workflow diagrams into a final end of year report.

References

1. Castle, E. (2003). Geodatabases in Design: A Floodpain Analysis of Little Kitten Creek. Thesis. Brigham Young University.
2. Rosenzweig, I. and B. Hodges, 2011. A Python Wrapper for Coupling Hydrodynamic and Oil Spill Models. Center for Research in Water Resources, University of Texas at Austin, CRWR Online Report 11-09. Submitted to Texas General Land Office Oil Spill Prevention & Response. FY 2011 Report under Contract No. 10-097-000-3928, October 31, 2011.
3. Robayo, O. and D. Maidment, 2005. Map to Map: Converting a NEXRAD Rainfall Map into a Flood Inundation Map. Center for Research in Water Resources, University of Texas at Austin, CRWR Online Report 05-1.
4. ESRI International User Conference Technical Workshops:
    a. "ModelBuilder Advanced Techniques," Scott Murrary, July 2010.
    b. "Working with Temporal Data in ArcGIS," David Kaiser, Hardeep Bajwa, July 2010.
5. ESRI Southeast Regional User Group (SERUG) Conference 2010 Technical Workshop: "Intermediate ModelBuilder," Kevin Armstrong.
6. Wikihow: "Creating time-series raster mosaics in ArcGIS 10 for Eye on Earth."
7. "ModelBuilder Lab," Geoinformatics, Spring 2008, Purdue University Library.
8. "Time-Series Contaminant Interpolation using ArcGIS and Spatial Analyst," Mark K. Petersen, ESRI User Conference Proceedings 2006, Paper 1326.

Appendix

Representative Python Scripts

mainORR.py

Imports the various Python libraries (in-built and customized) containing the scripts used for geoprocessing. Also sets the path to the data.

```
# ---------------------------------------------------------------------------
# Read the parameters
# Import arcpy and set path to data
# ---------------------------------------------------------------------------

import arcpy
import os
import time
import libFunctions
from libFunctions import *

------------------------------------------------------------------------------------------------------------------
```

def main():

Specifies the input and output parameters and the geoprocessing actions to be carried out.

```
  g.DEBUG = 0
  g.ARGIN = arcpy.GetParameterAsText(0)
  g.ARGA = arcpy.GetParameterAsText(1)
  g.ARGOUT = arcpy.GetParameterAsText(2)

  if g.ARGIN == '':
    iniInputFiles()
    ORR_clipSelected()

if __name__ == "__main__":
  main()
```

libFunctions.py

A customized library of Python scripts developed to automate geoprocessing activities that specifically support hydrological modeling work.

```
------------------------------------------------------------------------------------------------------------------
```

def iniInputFiles():

Gets the specified parameter by its index position from the list of parameters.

```
        result = []

## ARGIN = arcpy.GetParameterAsText(0)
        g.ARGIN =
"Z:/ORR_GIS/DOE_ORR_GeodB/Model_Builder/Model_Scripts/ORR_Project/DATA/TimeSe
ries.mdb"

## ARGA = arcpy.GetParameterAsText(1)
        g.ARGA =
"Z:/ORR_GIS/DOE_ORR_GeodB/Model_Builder/Model_Scripts/ORR_Project/DATA/TimeSe
ries.mdb/Model_Features/domain_EFPC"

## ARGOUT = arcpy.GetParameterAsText(2)
        g.ARGOUT =
"Z:/ORR_GIS/DOE_ORR_GeodB/Model_Builder/Model_Scripts/ORR_Project/DATA/TimeSe
ries.mdb"
```

----

def ORR_clipSelected():

Iterates the process of clipping features over a selected area (EFPC domain).

```
        CLIP_FEATURE = g.ARGA
        print "Iterating Selected Area:", CLIP_FEATURE, "\n"
        for inN in g.LVARN:
                print inN
                clipArea(inN, CLIP_FEATURE)
```

----

def clipArea(inN, ATLASSHEET):

Creates a unique scratch path name; Creates a temporary copy in memory; Checks for existence of data before deleting; Clips features; Lists the fields in output feature.

```
        sW = arcpy.env.scratchWorkspace = "in_memory"
        arcpy.overwriteOutput = True

        inF = g.ARGIN + "/" + inN
        outF = g.ARGOUT + "/CLIP_" + inN

  ## clipping is first used
  ##desc = arcpy.Describe(inF)
  ##oidName = str(desc.OIDFieldName)
```

```
    ##FTYPE = desc.ShapeType
    ## create temp copy in memory
    ##TMPFC = arcpy.CreateFeatureclass_management("in_memory","TMP",FTYPE,inF)
            TMP = sW + "/TMP"
    ##TMP = g.ARGOUT + "/TMP.shp"
             if arcpy.Exists(TMP):
        arcpy.Delete_management(TMP)
##    try:
##        arcpy.Delete_management(TMP)
##    finally:
##        print ""
             print " clipping: ", inF
            arcpy.Clip_analysis(inF,ATLASSHEET,TMP,"")
            ##output_fields = arcpy.ListFields(TMP)
            fieldnamesIN = [f.name for f in arcpy.ListFields(inF)]
            fieldnamesOUT = [f.name for f in arcpy.ListFields(TMP)]
            print "file IN=" , fieldnamesIN, " file OUT=" , fieldnamesOUT
    ## preferred use is to append the data to the existing file
            print " appending: ", outF
            if arcpy.Exists(outF):
                    arcpy.Append_management(TMP,outF,"TEST","","")
            else:
                    arcpy.CopyFeatures_management(TMP, outF)

    ##arcpy.Union_analysis (TMP, outF)
            if arcpy.Exists(TMP):
                    arcpy.Delete_management(TMP)
            arcpy.AddMessage(inN + " Done...")
```

----

def ListFeatureClasses():

Sets the workspace for the ListFeatureClass function; Uses the ListFeatureClasses function to return a list of shapefiles; Copies shapefiles to a geodatabase.

```
# Sets the workspace for the ListFeatureClass function
arcpy.env.workspace =
"Z:\ORR_GIS\DOE_ORR_GeodB\Model_Builder\Model_Scripts\ORR_Project\DATA\INPUT
DATA"

# Uses the ListFeatureClasses function to return a list of
#  shapefiles.
featureclasses = arcpy.ListFeatureClasses()

# Copy shapefiles to a geodatabase
for fc in featureclasses:
   arcpy.CopyFeatures_management(
      fc,
os.path.join("Z:\ORR_GIS\DOE_ORR_GeodB\Model_Builder\Model_Scripts\ORR_Project\D
ATA\TimeSeries.mdb",
                     os.path.splitext(fc)[0]))
```

---

def AddLayer():

Adds a new layer from a layer (.lyr) file on disk and places it at the bottom of the data frame called EFPC_Data.

```
import arcpy
mxd =
arcpy.mapping.MapDocument(r"Z:\ORR_GIS\DOE_ORR_GeodB\Model_Builder\Model_Scri
pts\ORR_Project\MAP.mxd")
df = arcpy.mapping.ListDataFrames(mxd, "EFPC_Data")[0]
addLayer =
arcpy.mapping.Layer(r"Z:\ORR_GIS\DOE_ORR_GeodB\Model_Builder\Model_Scripts\ORR_
Project\DATA\INPUTDATA\stations_OREIS_mercury.lyr")
arcpy.mapping.AddLayer(df, addLayer, "BOTTOM")
mxd.saveACopy(r"Z:\ORR_GIS\DOE_ORR_GeodB\Model_Builder\Model_Scripts\ORR_Proje
ct\DATA\OUTPUTDATA\ORR_Project.mxd")
del mxd, addLayer
```

TS_InterpolateToRaster_Script_080912.py

Interpolates GW elevation timeseries data from wells within Y-12 area to create rasters by date and exports map in PDF format from ArcMap document.

```
# ---------------------------------------------------------------------------
# TS_InterpolateToRaster_Script_080912.py
# Created on: 2012-08-09 09:40:35.00000
#   (generated by ArcGIS/ModelBuilder)
# Description: Interpolates GW elevation timeseries data from wells within Y-12 area to
create rasters by date and exports map in PDF format from ArcMap document.
# ---------------------------------------------------------------------------

# Import arcpy module
import arcpy

# Check out any necessary licenses
arcpy.CheckOutExtension("spatial")

# Load required toolboxes
```

```
arcpy.ImportToolbox("Model Functions")
arcpy.ImportToolbox("C:/Documents and Settings/lawrence/Application
Data/ESRI/Desktop10.0/ArcToolbox/My Toolboxes/ORR_GeodB.tbx")

# Local variables:
GW_ELEV_TS = "GW_ELEV_TS"
ORR_GeodB_Scratch_mdb = "C:\\Documents and Settings\\lawrence\\My
Documents\\ArcGIS\\ORR_GeodB_Scratch.mdb"
RasterSeries__2_ =
"Z:\\ORR_GIS\\DOE_ORR_GeodB\\Model_Builder\\TimeSeries.mdb\\RasterSeries"
Model_Builder_Testing_mxd =
"Z:\\ORR_GIS\\DOE_ORR_GeodB\\Model_Builder\\Model_Builder_Testing.mxd"
I_GW_ELEV_TS_DATE_ = "I_GW_ELEV_TS_DATE_"
GW_ELEV_n_ = "C:\\Documents and Settings\\lawrence\\My
Documents\\ArcGIS\\ORR_GeodB_Scratch.mdb\\GW_ELEV%n%"
GW_ELEV_n__tif =
"Z:\\ORR_GIS\\DOE_ORR_GeodB\\Model_Builder\\Raster_TS_Outputs\\GW_ELEV%n%.tif
"

# Process: Iterate Feature Selection
arcpy.IterateFeatureSelection_mb(GW_ELEV_TS, "DATE_ #", "false")

# Process: Natural Neighbor
arcpy.gp.NaturalNeighbor_sa(I_GW_ELEV_TS_DATE_, "GW_ELEV", GW_ELEV_n_,
"0.222827600002289")

# Process: Workspace To Raster Catalog
arcpy.WorkspaceToRasterCatalog_management(ORR_GeodB_Scratch_mdb,
RasterSeries__2_, "NONE", "NONE")

# Process: Copy Raster
arcpy.CopyRaster_management(GW_ELEV_n_, GW_ELEV_n__tif, "", "", "", "NONE", "NONE",
"")

# Process: ExportToPDF Script
arcpy.gp.toolbox = "C:/Documents and Settings/lawrence/Application
Data/ESRI/Desktop10.0/ArcToolbox/My Toolboxes/ORR_GeodB.tbx";
# And replace arcpy.gp.ExportToPDF(...) with arcpy.ExportToPDF_ALIAS(...)
arcpy.gp.ExportToPDF()
```

SelectData_ByAttribute.py

Selects GW wells by station name and exports all associated attributes in a .dbf table.

```
# ---------------------------------------------------------------------------
# SelectData_ByAttribute.py
# Created on: 2013-01-09 10:49:53.00000
#   (generated by ArcGIS/ModelBuilder)
# Usage: SelectData_ByAttribute <Value> <GW_ELEV_TS> <GW_ELEV_TS__2_>
# Description:
# This model selects GW wells by station name and exports all associated attributes in a
.dbf table.
# ---------------------------------------------------------------------------

# Import arcpy module
import arcpy

# Load required toolboxes
arcpy.ImportToolbox("Model Functions")

# Script arguments
Value = arcpy.GetParameterAsText(0)
if Value == '#' or not Value:
    Value = "1" # provide a default value if unspecified

GW_ELEV_TS = arcpy.GetParameterAsText(1)
if GW_ELEV_TS == '#' or not GW_ELEV_TS:
    GW_ELEV_TS = "GW_ELEV_TS" # provide a default value if unspecified

GW_ELEV_TS__2_ = arcpy.GetParameterAsText(2)
if GW_ELEV_TS__2_ == '#' or not GW_ELEV_TS__2_:
    GW_ELEV_TS__2_ = "GW_ELEV_TS" # provide a default value if unspecified

# Local variables:
I_GW_ELEV_TS = GW_ELEV_TS__2_
QueryTable_Value__dbf = I_GW_ELEV_TS
QueryTable_Value__Statistics_dbf = QueryTable_Value__dbf

# Process: Select Layer By Attribute
arcpy.SelectLayerByAttribute_management(GW_ELEV_TS, "NEW_SELECTION", "")

# Process: Iterate Feature Selection
arcpy.IterateFeatureSelection_mb(GW_ELEV_TS__2_, "", "false")

# Process: Copy Rows
arcpy.CopyRows_management(I_GW_ELEV_TS, QueryTable_Value__dbf, "")

# Process: Summary Statistics
arcpy.Statistics_analysis(QueryTable_Value__dbf, QueryTable_Value__Statistics_dbf,
"GW_ELEV MEAN;GW_ELEV MIN;GW_ELEV MAX;GW_ELEV RANGE;GW_ELEV
STD;GW_ELEV COUNT", "")
```

Sample ArcGIS ModelBuilder Process Workflow Diagram

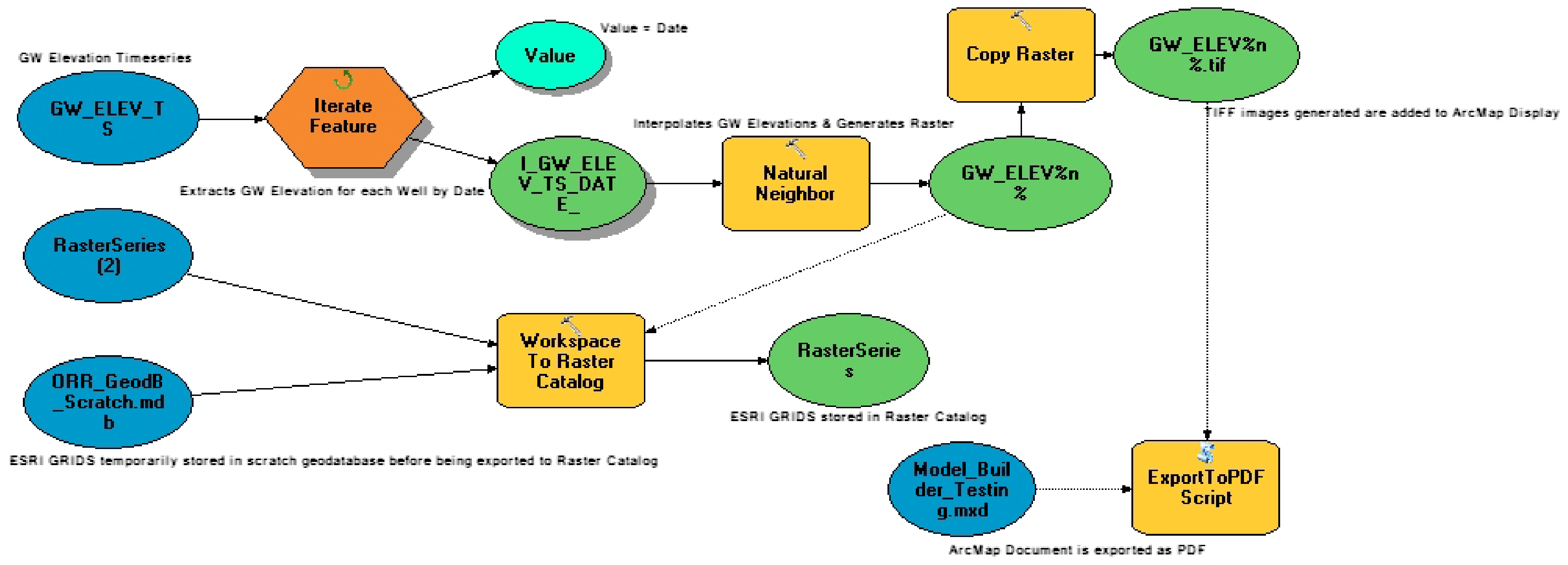
GW Elevation Timeseries
GW_ELEV_TS
Iterate Feature
Value
Value = Date
I_GW_ELEV_TS_DATE_
Extracts GW Elevation for each Well by Date
Interpolates GW Elevations & Generates Raster
Natural Neighbor
GW_ELEV%n%
Copy Raster
GW_ELEV%n%.tif
TIFF images generated are added to ArcMap Display
RasterSeries (2)
Workspace To Raster Catalog
RasterSeries
ESRI GRIDS stored in Raster Catalog
ORR_GeodB_Scratch.mdb
ESRI GRIDS temporarily stored in scratch geodatabase before being exported to Raster Catalog
Model_Builder_Testing.mxd
ExportToPDF Script
ArcMap Document is exported as PDF

Figure 5. Process workflow diagram created using ArcGIS ModelBuilder. This model interpolates groundwater elevation timeseries to create rasters by date and exports maps in PDF format.

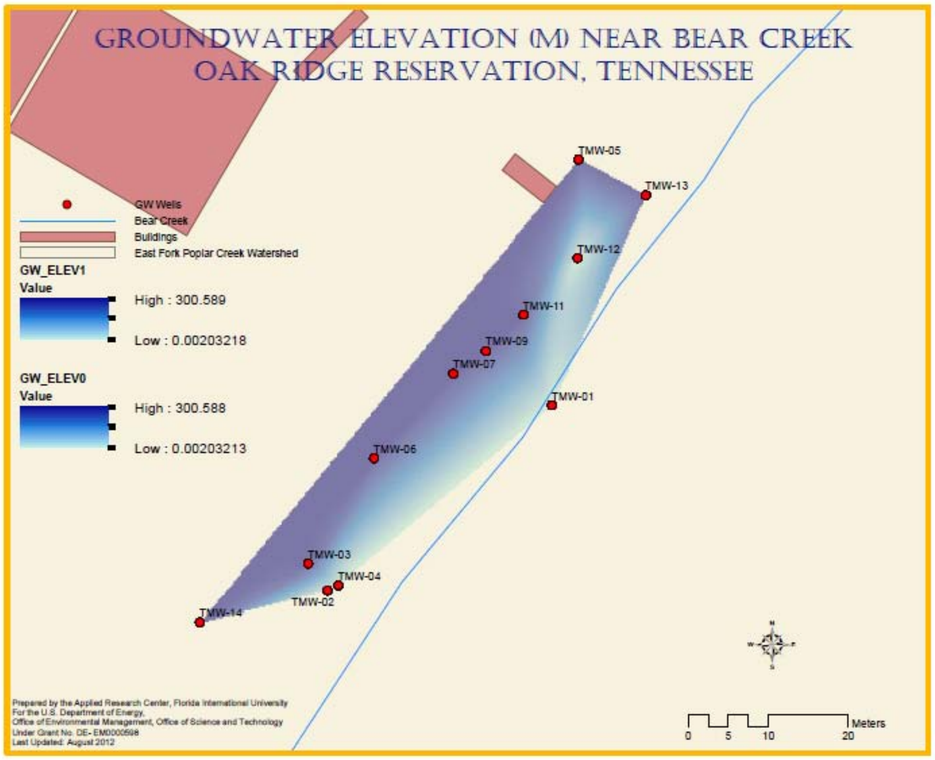
GROUNDWATER ELEVATION (M) NEAR BEAR CREEK
OAK RIDGE RESERVATION, TENNESSEE
GW Wells
Bear Creek
Buildings
East Fork Poplar Creek Watershed
GW_ELEV1
Value
High : 300.589
Low : 0.00203218
GW_ELEV0
Value
High : 300.588
Low : 0.00203213
TMW-05
TMW-13
TMW-12
TMW-11
TMW-09
TMW-07
TMW-01
TMW-06
TMW-03
TMW-04
TMW-02
TMW-14
Prepared by the Applied Research Center, Florida International University
For the U.S. Department of Energy,
Office of Environmental Management, Office of Science and Technology
Under Grant No. DE- EM0000598
Last Updated: August 2012
Meters
0
5
10
20

Figure 6. Map generated in PDF format following interpolation of GW elevation data.