Brigham Young University
BYU ScholarsArchive

Theses and Dissertations

2005-03-16

# Floodplain and Flood Probability Mapping Using Geodatabases

Douglas J. Gallup
*Brigham Young University - Provo*

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

 Part of the Civil and Environmental Engineering Commons

BYU ScholarsArchive Citation
Gallup, Douglas J., "Floodplain and Flood Probability Mapping Using Geodatabases" (2005). *Theses and Dissertations*. 262.
https://scholarsarchive.byu.edu/etd/262

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

www.manaraa.com

# FLOODPLAIN AND FLOOD PROBABILITY MAPPING USING GEODATABASES

by

Douglas J. Gallup

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

April 2005

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

Copyright © 2005 Douglas J. Gallup

All Rights Reserved

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Douglas J. Gallup

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

| | |
|---|---|
| Date | E. James Nelson, Chair |
| Date | Alan K. Zundel |
| Date | Norman L. Jones |

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Douglas J. Gallup in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

E. James Nelson
Chair, Graduate Committee

Accepted for the Department

A. Woodruff Miller
Department Chair

Accepted for the College

Douglas M. Chabries
Dean, Ira A. Fulton College of Engineering and Technology

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

ABSTRACT

FLOODPLAIN AND FLOOD PROBABILITY
MAPPING USING GEODATABASES

Douglas J. Gallup
Department of Civil and Environmental Engineering
Master of Science

This research presents methods of creating digital maps for floodplain delineation and flood probability studies and storing them in a geodatabase. Methods for creating a geodatabase for water resources outside of a GIS are presented. The geodatabase follows the ArcHydro data model. Methods are also shown for creating digital flood maps and storing them in the geodatabase. These flood maps, defining the floodplain boundary and flood probability, are stored in a digital format ready for use in a FEMA flood hazard project, allowing for better archival methods and reproducibility.

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

# ACKNOWLEDGMENTS

Many people have helped me out during my graduate research, and I am grateful for them. I would like to thank my advisor, Dr. E. James Nelson, for his help and guidance throughout my research and writing and especially for his guidance and encouragement. I would also like to thank my co-workers for their assistance. Finally, I wish to thank my family, for all their support throughout the years and for the value they've placed on education.

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

# Table of Contents





www.manaraa.com

المنارة للاستشارات
www.manaraa.com

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

## List of Tables





www.manaraa.com

المنارة للاستشارات
www.manaraa.com

## List of Figures





المنارة للاستشارات
www.manaraa.com

المنارة للاستشارات
www.manaraa.com

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

# 1 Introduction

The acquisition of data for developing hydrologic and floodplain models, along with storing it in a common format that is readily shareable and readable has always been an issue to water resources engineers. Different computer programs and models exists, but with their own data specifications and formats.

The Environmental Modeling Research Laboratory at Brigham Young University, with its own data formats in its Watershed Modeling System (WMS) software, desired to be able to share its model data and results in a common, industry standard format. The ArcObjects tools, developed by Environmental Modeling Research Institute (ESRI), were looked at as a means of developing and storing hydrologic data in an easy to use system.

The objective of this research was to see if ArcObjects could be implemented within WMS to manage and store hydrologic, hydraulic and floodplain modeling data. ArcObjects were developed primarily to extend the ArcView suite of GIS software. This research, was initiated to see whether using ArcObjects in a separate program from ArcView was feasible, and shows how to take spatial and non-spatial data outside of a GIS to build hydrologic, hydraulic and floodplain models within a geodatabase. Though implemented in (WMS), the methods presented show how to take generic data representing watersheds, channels and floodplains from any source outside of a GIS for storage in a geodatabase.

A method of managing and storing hydrologic, hydraulic and floodplain modeling data is the first result of this research. A geodatabase is a relational database used to store both geographic coordinates and attribute data. As opposed to other methods of storing data, the geodatabase has built in capabilities for creating relationships between spatial data,



www.manaraa.com

between non-spatial attributes, and between spatial and non-spatial data. ArcObjects are used to create the geodatabase, following the ArcHydro data model.

In addition, this research presents a method of storing digital flood maps in a geodatabase. Rather than only dealing with floodplain maps of a given recurrence interval, methods of storing maps taking into account uncertainty in the modeling process are also included. These floodplain maps are compatible with the ArcHydro data model, and in a format ready for creating digital flood insurance rate maps for the Federal Emergency Management Agency's flood hazard mapping projects.

This thesis discusses geodatabases and the storage of digital flood maps, so that the results of hydrologic, hydraulic and floodplain delineation models can be placed in an industry standard format. With the floodplain maps inside a geodatabase, the data can be used both by engineers running the model as well as by the government agencies responsible for creating policy and establishing insurance rates. When stored as a geodatabase, these digital maps have advantages over typical paper maps, or paper maps that have been converted to a digital format by scanning into a computer.

Section 2 discusses hydrologic, hydraulic and floodplain delineation models. It covers the data required to run these models, and how to create maps such as flood extent maps and flood probability maps. The methods for creating digital flood insurance rate maps for FEMA projects are also covered. The geodatabase model is described, which allows relationships between both tabular as well as geographic data. The data that can be stored in a geodatabase are also discussed, including the spatial data required in a FEMA project.

Section 3 outlines the method of taking spatial data and storing it in a geodatabase in the ArcHydro format. Computer programming code and ArcObjects are used to create the geodatabase, and extend the ArcHydro data model by including flood maps and digital terrain data.

Section 4 includes the conclusions from this research. This section outlines the contributions to floodplain mapping and presents ideas for future research.



www.manaraa.com

# 2 Floodplain Delineation and Mapping

In order to produce maps suitable for evaluation of flood insurance rates and flooding hazards, proper methods of hydrologic and hydraulic analysis along with floodplain delineation must be performed. This chapter presents the models and data used to develop floodplain and flood probability maps, to be stored in geodatabases or to generate digital maps to be used in FEMA studies.

## 2.1 Hydrologic and Hydraulic Analysis

Two common hydrologic and hydraulic analysis packages for floodplain modeling are HEC-1 (US Army Corps of Engineers Hydrologic Engineering Center, 1998) and HEC-RAS (US Army Corps of Engineers Hydrologic Engineering Center, 2001). These or other similar models are used to develop the necessary data required to run a floodplain delineation model.

### 2.1.1 Hydrologic model

HEC-1, developed by the Hydrologic Engineering Center, "simulates the surface runoff response of a river basin to precipitation by representing the basin as an interconnected system of hydrologic and hydraulic components" (US Army Corps of Engineers Hydrologic Engineering Center, 1998). As a lumped parameter model, the watershed is organized into a connected series of reaches and basins, along with other units. Basic parameters for this model are shown in Table 2.1.

The result of a HEC-1 model will be the response of the watershed to an input rainfall storm and physical parameters of the watershed. These parameters can be varied to develop hydrographs for different flood recurrence intervals. WMS (Environmental Modeling



www.manaraa.com

Table 2.1: Basic input parameters for HEC-1.

| **Basin** | **Precipitation** | **Losses** | **Unit Hydrograph** | **Routing** |
|---|---|---|---|---|
| Area | Storm Total | Curve Number | Unit Hydrograph | Method |
| Baseflow | Distribution | Green-Ampt | TC or Lag Time | Storage |

Research Laboratory, 2003) and other GIS programs like HEC-GeoHMS (US Army Corps of Engineers Hydrologic Engineering Center, 2003) process digital data as a preprocessor to HEC-1.

### 2.1.2 Hydraulic model

HEC-RAS, also developed by the Hydrologic Engineering Center, is a hydraulic model based on the one dimensional energy equation, and includes steady state and unsteady models (US Army Corps of Engineers Hydrologic Engineering Center, 2001). In the steady state model, a set of geometric and flow parameters are used to set up the model, as shown in Table 2.2.

The result of a HEC-RAS model is a water surface elevation at each cross section. The model parameters, such as the initial flow in the channel, are varied in order to develop

Table 2.2: Basic input parameters for HEC-RAS.

| **Geometric** | **Flow (at channel boundaries)** |
|---|---|
| Cross section stations | Channel flow |
| Cross section geometry | Channel slope |
| Manning's roughness | Water depth |
| | Rating curve |



www.manaraa.com

different sets of water surface elevations for different flood recurrence intervals. GIS may be used as a preprocessor for HEC-RAS and has the ability to edit the geometric data such as cross sections, and to develop the hydraulic parameters (Noman, 2001).

## 2.2 Floodplain Delineation

Automated floodplain delineation is an excellent tool for producing flood maps. Given the right hydrologic and hydraulic information, along with digital elevation data, channel cross sections and flow values, a floodplain can be generated. Automated methods can perform more calculations with larger sets of data in order to determine the extent and depth of flooding. Floodplain maps created from these studies may be stored in a digital format such as a geodatabase, in order to create flood insurance rate maps and develop plans that help minimize flood damage.

### 2.2.1 Floodplain models

While many methods of delineating a floodplain exist (Bedient & Huber, 1998), this thesis focuses on the work of Noman (2001), in the Watershed Modeling System (WMS). The model is used to generate flood maps of any recurrence interval based on a set of water surface elevations, either from observed data or from the result of hydrologic and hydraulic modeling.

The model as described by Noman (2001) requires:

- A digital terrain model of the floodplain area
- A set of water surface elevations
- Optionally, a set of flood barriers

The digital terrain model is a set of elevation data used to determine the floodplain, while the water levels are usually derived from a hydraulic model, such as HEC-RAS. The elevation data formats include grids, such as a USGS DEMs, light detection and radar (LIDAR)



www.manaraa.com

data, a triangulated irregular networks (TIN), sets of contour lines, or cross sections. Flood barriers are used in the absence of detailed digital elevation to determine areas which are not flooded, such as regions protected by levees. If desired, a set of flood barrier data are also used. Flood barriers are stored as polylines or polygons and are used to protect areas such as levees, whether represented in the DTM or not, which are barriers to the spread of a flood.

Next, water surface levels are computed and combined with the DTM. These values are the result of a hydraulic model like HEC-RAS, and are stored as an XY scattered dataset in WMS. Multiple datasets store different water surface elevations at each point.

According to the algorithm developed by Noman (2001), water surface elevations resulting from the hydraulic model are interpolated throughout the DTM. This results in a flood depth dataset, which is used to create a flood depth map, a flood extent map and a flood impact map. An area is considered flooded wherever the flood depth is greater than the area's ground elevation. The flood extent map is formed wherever the water surface level is equal to the ground elevation, as shown in Figure 2.1.

The floodplain delineation model is then repeated in order to develop flood maps for multiple recurrence intervals. These maps are then exported in electronic format for developing FEMA digital flood insurance rate maps (DFIRMs).

### 2.2.2 Stochastic floodplain modeling in WMS

Smemoe (2004) has presented another method of generating floodplains based on a stochastic approach. Using the same floodplain delineation model as Noman (2001), this new method produces flood probability and annual exceedance probability (AEP) maps.

The flood probability map takes into account the natural variability and model uncertainty by creating a map showing the probability of flooding for a given recurrence interval. The AEP map takes into account the same variability and uncertainty and shows the probability of a point on the map being flooded in any given year. These maps are created



www.manaraa.com

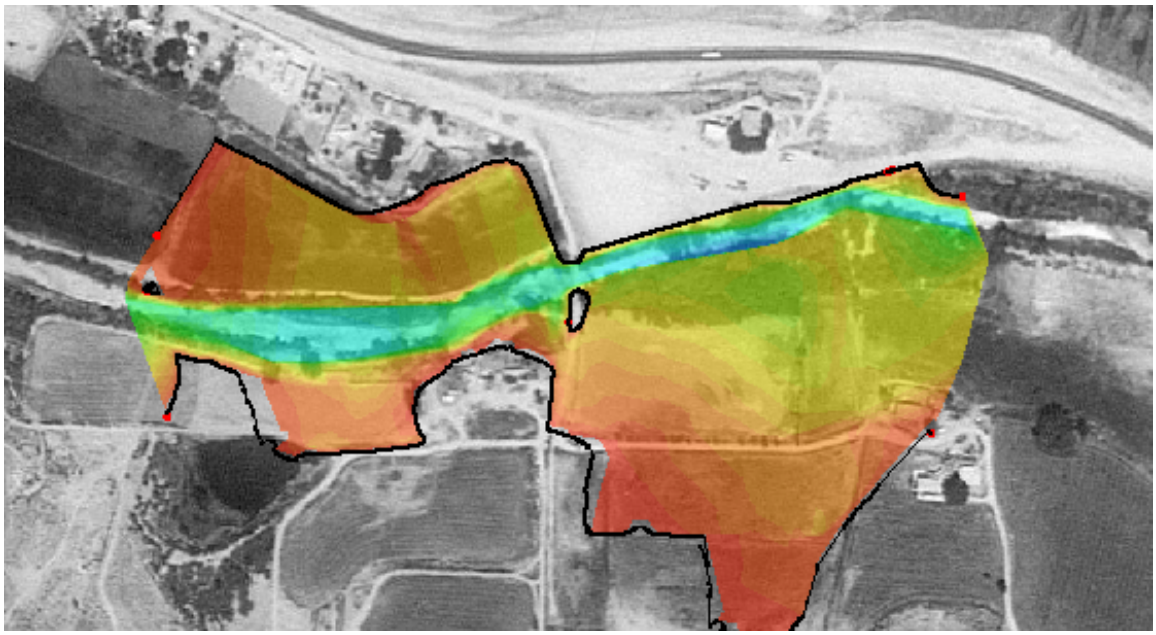


Figure 2.1: A flood extent map created in WMS.

by running the hydrologic, hydraulic and floodplain models tens or hundreds of times and varying the input parameters using stochastic modeling techniques each time. Parameters such as rainfall and the curve number for computing losses in HEC-1 are given a mean value and a standard deviation. Then, a Monte Carlo simulation is run, generating different parameter sets for each instance based on the probability distribution functions of the stochastic parameters set up by the model. Given enough model runs, this method shows the uncertainty in modeling a floodplain boundary based on the variability of the input parameters and the modeling process. Figures 2.2 and 2.3 are examples of a flood probability and AEP map. Smemoe (2004) further defines the methodologies used to develop the maps.

While not yet accepted as a standard, the flood probability maps for different recurrence intervals and the AEP could be used for developing FEMA flood insurance rate maps. In addition to the standard flood extent maps, the flood probability and AEP maps provide a better understanding of determining a floodplain and the uncertainty involved. The results



www.manaraa.com

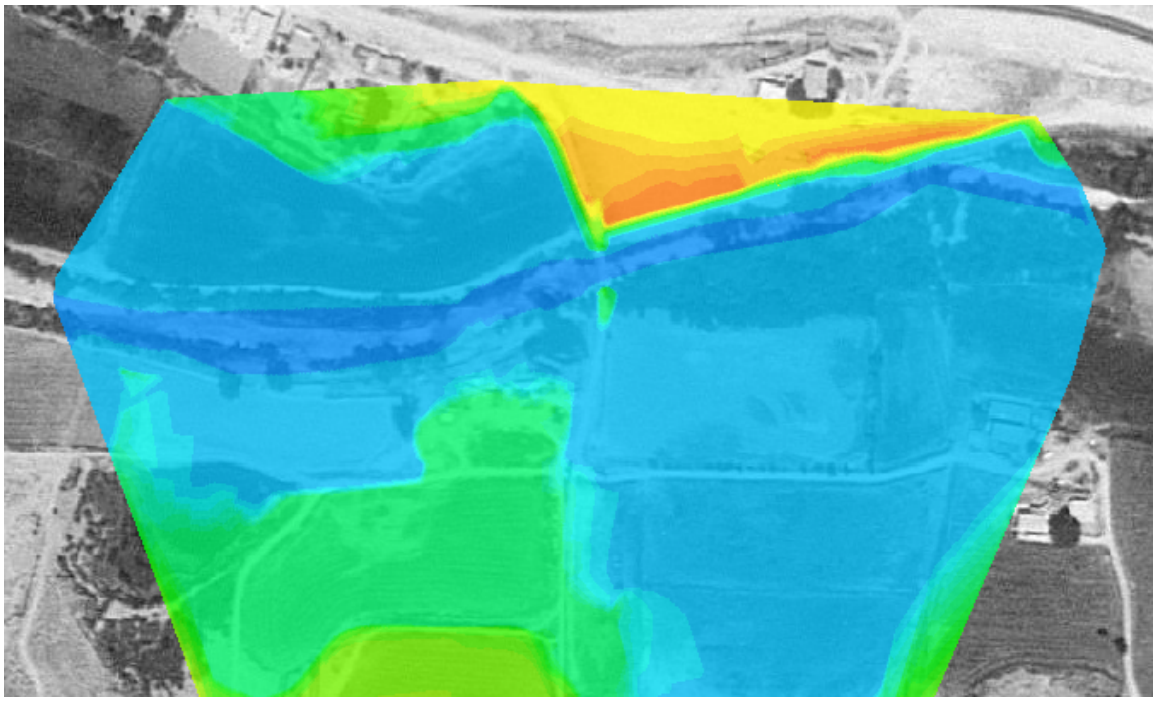


Figure 2.2: A flood probability map created in WMS.

of the floodplain delineation and flood probability studies are stored and managed in a GIS and are used to generate DFIRMs and other digital maps.

## 2.3 Digital Flood Insurance Rate Maps

As part of the National Flood Insurance Act of 1968, the Federal Emergency Management Agency (FEMA) has specific mandates to identify flood hazards as part of the National Flood Insurance Program (Federal Emergency Management Agency, 2003). Part of FEMA's duties include creating and maintaining flood insurance rate maps, which are stored in digital formats.

### 2.3.1 Creating FEMA Maps

In order to create a new FIRM or update an existing FIRM, FEMA follows a four step process (Federal Emergency Management Agency, 2003):



www.manaraa.com

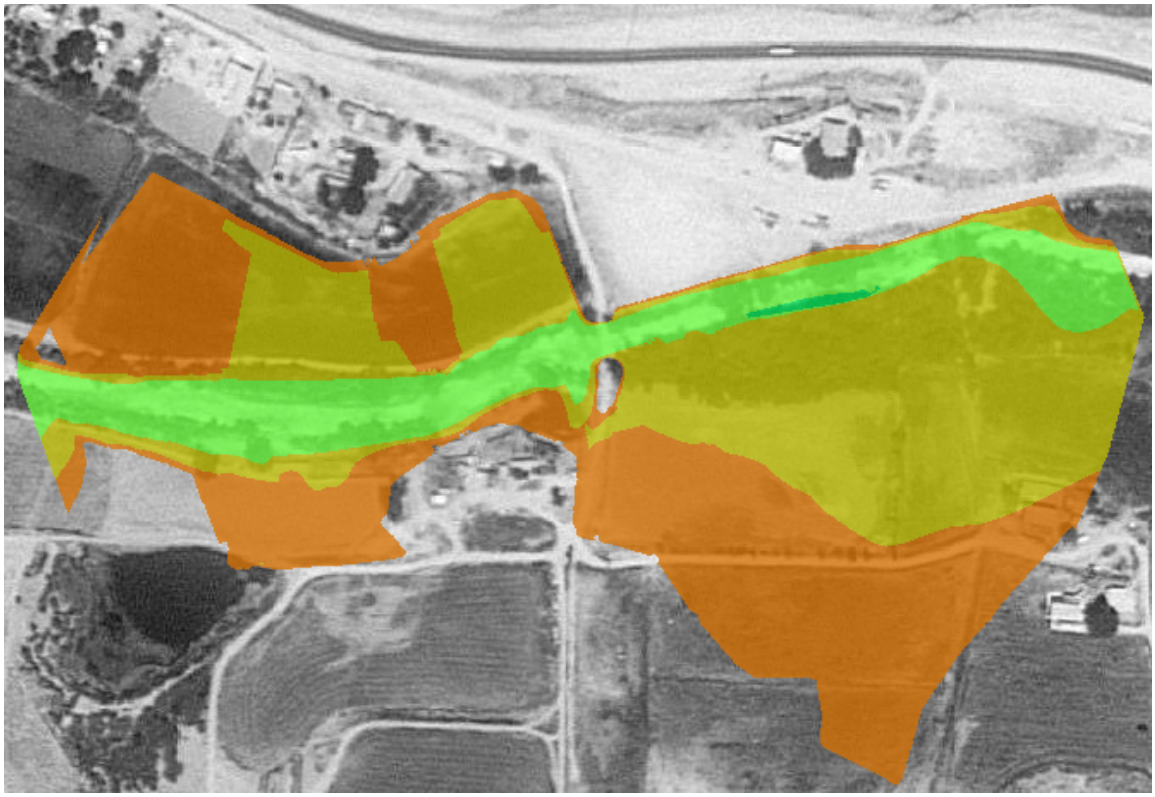


Figure 2.3: An AEP map created in WMS.



www.manaraa.com

1. Mapping needs assessment

2. Project scope

3. Topographic and flood hazard data development/report and map production

4. Preliminary/post-preliminary processing

In the first step, mapping needs assessment, FEMA evaluates the need for updating current flood maps or creating new maps in areas without any data. This involves comparing the existing maps with new data from observed flood events, changes to the watershed such as landuse changes or newly constructed flood control structures, changes in the stream morphology, and other factors that would affect the hydrologic, hydraulic and flood models.

The project scoping process includes identifying the necessary data and research to be used to update or create the flood maps, and making decisions on the format of the resulting FIRM, such as whether it will be digital or not. Scoping also involves organizing the project team for performing the study.

To develop the topographic and flood hazard data, field surveys are performed. These include collecting data from stream cross sections, hydraulic structures and aerial surveys. Based on this topographic data, a new floodplain boundary is established along with digital flood hazard maps.

Once the reports are written and the flood maps are created, the preliminary processing and post-preliminary processing occurs. In these steps, the results of the project are distributed and presented to the affected communities. The FIRM becomes open to review and outside analysis, for the purpose of possibly changing the floodplain boundaries.

The first FIRMs were produced and published on paper. These maps were first put into digital format by scanning them into a computer image, such as shown in Figure 2.4, allowing the results to be shared through the internet. A limitation of the scanned imaged, however, is that it does not contain the model input and output data used in the study, only a visual representation of the results. A digital FIRM, such as the one shown in



www.manaraa.com

Figure 2.5 (Federal Emergency Management Agency Map Service Center, 2004), holds not only the final floodplain boundaries, but the model parameters and data collected to perform the analysis. This map, for example shows flood hazard areas as classified by the project, while the data used to build the map is stored on a computer in a geodatabase.

## 2.4 History

Geodatabases, or geographic databases, are the state of the art technology for storing geographical data. Geodatabases are used as part of geographic information systems to make intelligent decisions over a wide variety of fields, including water resources. While the concept of geodatabases is relatively new, they are viewed as the successors to other forms of storing geographic data and its associated attributes. These other formats include CAD data, databases and coverages.

### 2.4.1 Early data models

Some of the first computerized maps were produced through CAD computer software. While not as rich in features and capabilities compared to the modern GIS software, CAD played an important role in decision making through storing vector data such as points, lines and polygons, as shown in Figure 2.6. While these shapes still make up the basis of the information stored in a modern GIS, the early CAD software had the limitation of not being able to store attribute information , such as that found in a database (Zeiler, 1999). Information inside of a database is queried in order to aid in analysis or to extract important information from a part of the database.

An improved geographic model, the coverage data model, was introduced in 1981 with the release of ArcInfo by Environmental Systems Research Institute, Inc. In addition to storing the same point, polyline and polygon information found in the CAD data model, the coverage model provided attribute tables to store non-spatial data in an expanded regular format, as shown in Figure 2.7. Further, the coverage model also introduced topological



www.manaraa.com

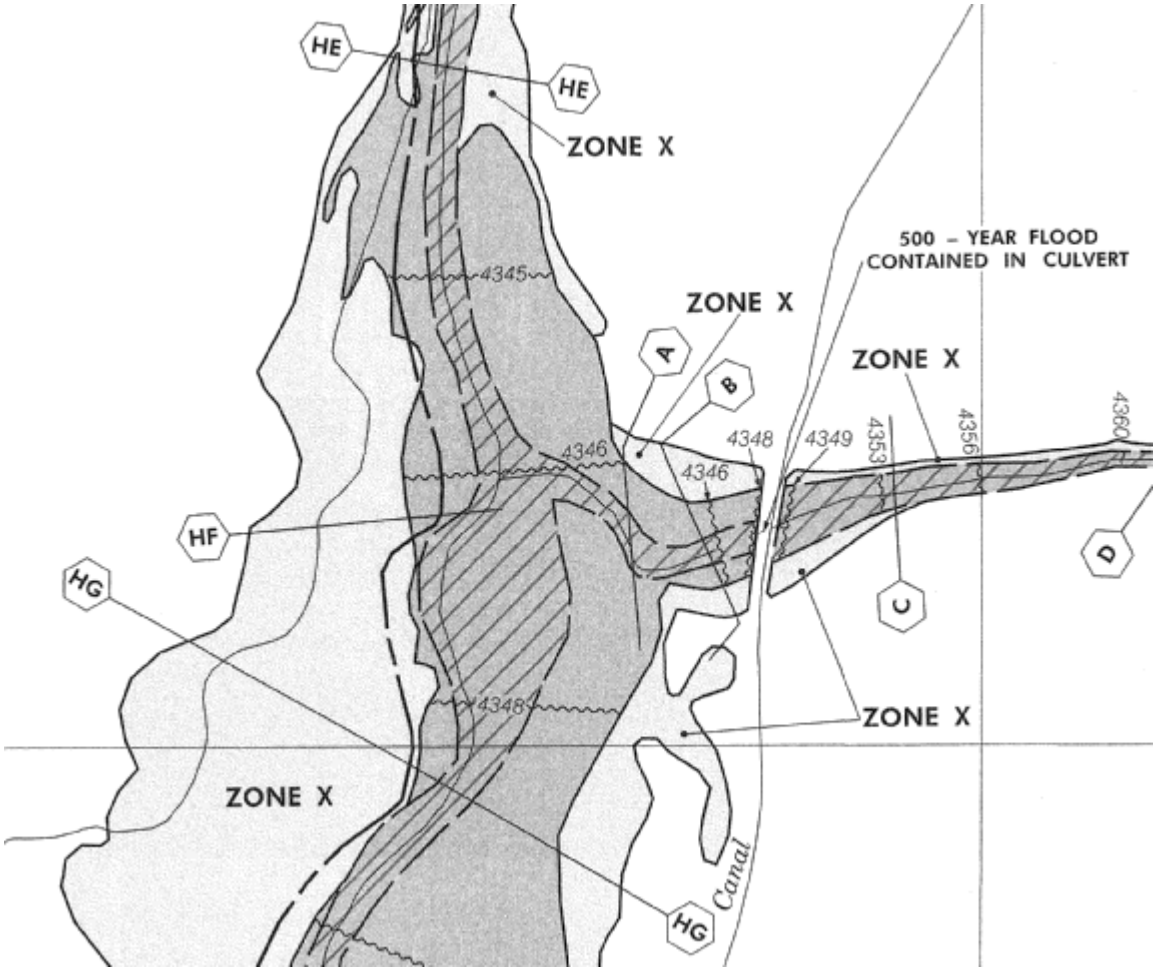



Figure 2.4: A paper FIRM scanned into a computer image format.



www.manaraa.com

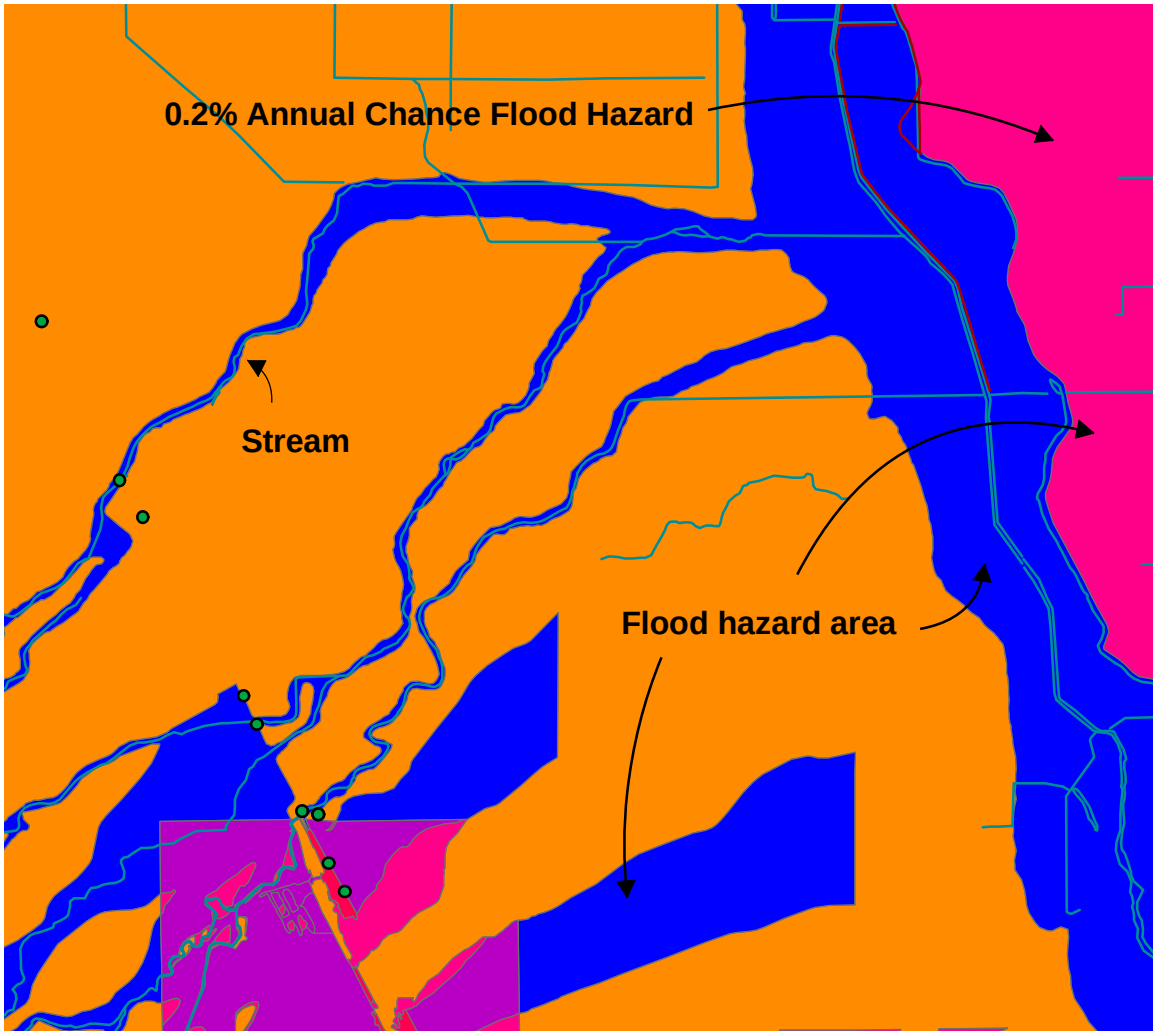



Figure 2.5: DFIRM from a geodatabase for Colusa County, CA, showing flood hazards.



www.manaraa.com

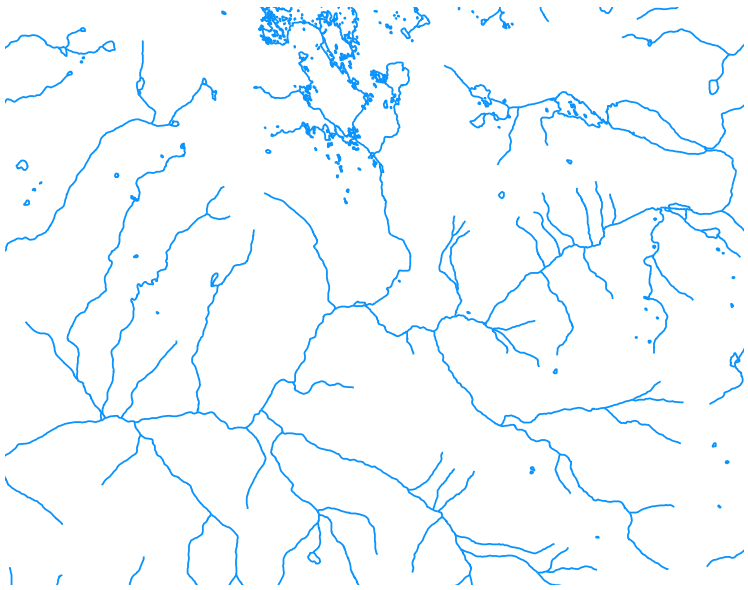


Figure 2.6: Streams as CAD data.

relationships, which allowed the GIS software to easily determine connecting or adjacent features, such as the upstream and downstream polylines connected to any given polyline. The vector data, consisting of direction, magnitude and connectivity data, were stored in indexed binary files while the attribute tables were stored in a database (Zeiler, 1999).

Despites these advances over the CAD model, the coverage model also has some limitations in describing the behavior of some objects. When modeling streams, for example, they

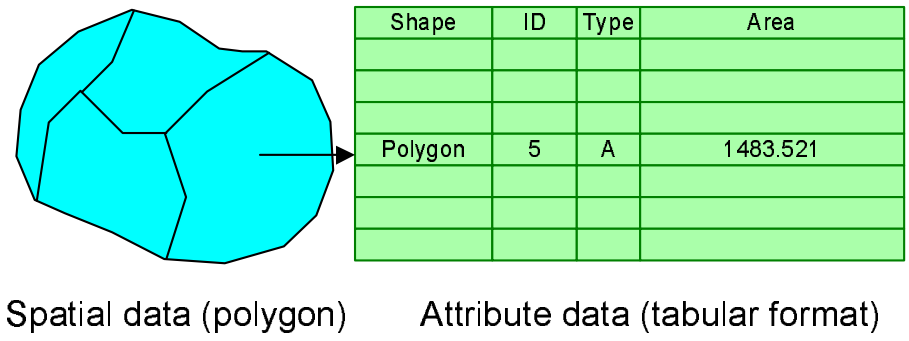



Figure 2.7: Spatial and non-spatial data in a coverage.



www.manaraa.com

should have a certain direction (flowing downhill) and should converge with other streams as the features meet, and combine their flows. While the two streams might be linked topologically in the coverage model, their behavior at the junction is undefined (Zeiler, 1999). Further, the different types of features (points, lines and polygons) are separated from one another in the coverage model, when in fact the outlet (point feature) of a drainage area (polygon feature) on a stream (line feature) are all related topologically and behaviorally. In the coverage model, the separated map data allows for easier management and map production, as each set of data can be edited or updated without the other. However, when dealing with watershed modeling applications, the spatial and attribute data needs to be connected and related to each other, such as a stream arc needing to know which drainage area it is located in. Figure 2.8 shows the background elevation data and vector data used to describe a watershed. The background elevation could be in the form of raster or gridded data, or as triangulated irregular networks (TINs).

## 2.5 The Geodatabase Model

Geodatabases first came out with the release of ArcGIS 8, by ESRI (2000). The geodatabase model builds on the coverage model by adding new capabilities such as spatial relationships between layers. Unlike the previous coverage data model, the geodatabase model is object-oriented, meaning that map entities are able to be managed as groups of points, lines and polygons instead of individually. Other behaviors of these objects such as levees and flood barriers are be defined, giving the geodatabase a much greater flexibility when modeling complex environments (Zeiler, 1999). In addition, it includes built in tools to define how different objects interact with one another.

### 2.5.1 New capabilities

While the coverage model provided for topological relationships, the geodatabase model also adds spatial and general relationships to be made between objects. Topological relationships



www.manaraa.com

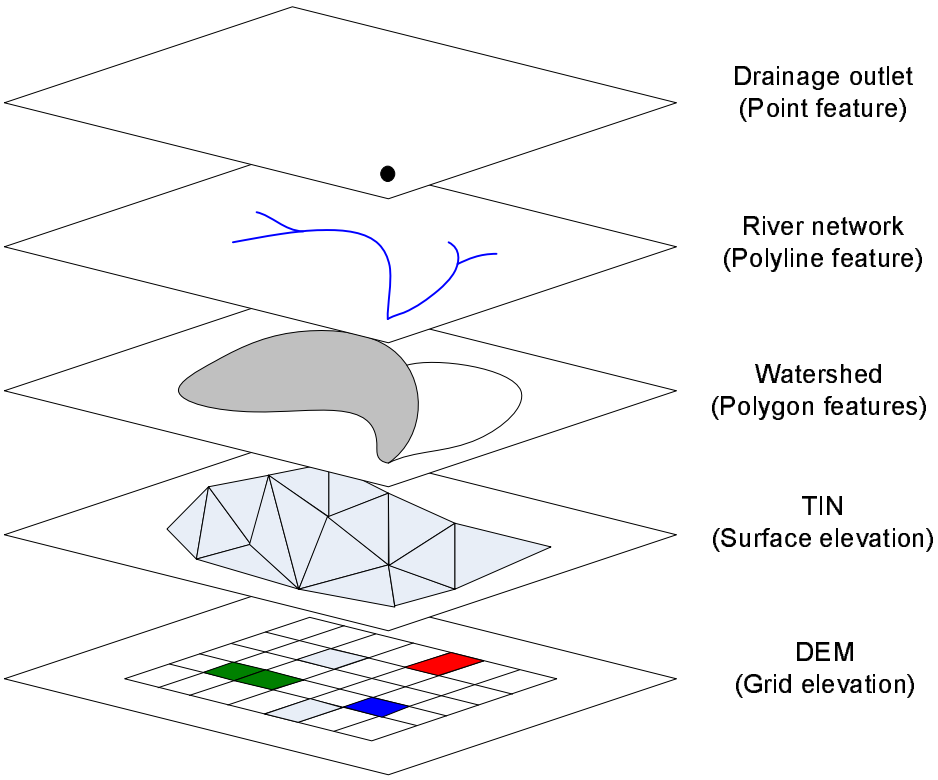



Figure 2.8: Watershed data composition.



www.manaraa.com

are used to ensure that streams are connected to one another as they progressively branch out upstream. Spatial relationships can be used to determine which watershed a stream is located in. General relationships are established when it is not clear whether two objects are spatially related, such as a monitoring point located on a stream at the confluence of another stream. Validation rules are created to ensure proper relationships and connectivity between data (Zeiler, 1999).

### 2.5.2 The composition of a geodatabase

As illustrated in Figure 2.9, a geodatabase consists of for main types of data:

- Vector data (points, polylines and polygons) and its associated attributes
- Raster or gridded data
- Triangulated irregular networks (TINs)
- Locators

These data are typically stored inside datasets within the geodatabase, which allows them to be related to each other (Zeiler, 2001). Datasets are used to store groups of related data for a model, such as all the necessary data to characterize a watershed.

Vector data are stored in the geodatabase in groupings called feature datasets. Each feature dataset must have a spatial reference defined for it (such as a projected or geographic coordinate system), which is common to each set of vector data. The most common type of data stored in a feature dataset are feature classes. A single feature class stores only one type of vector data (such as points, lines, polygons or annotations), along with its associated attributes which are stored as a table. This is different than the coverage model, where each set of vector data is stored in its own folder, with its attribute table stored separately in another folder as part of the coverage framework.

A feature dataset can also store object classes. An object class is similar to a feature class, except that it does not hold any spatial information (such as vector data) in its



www.manaraa.com

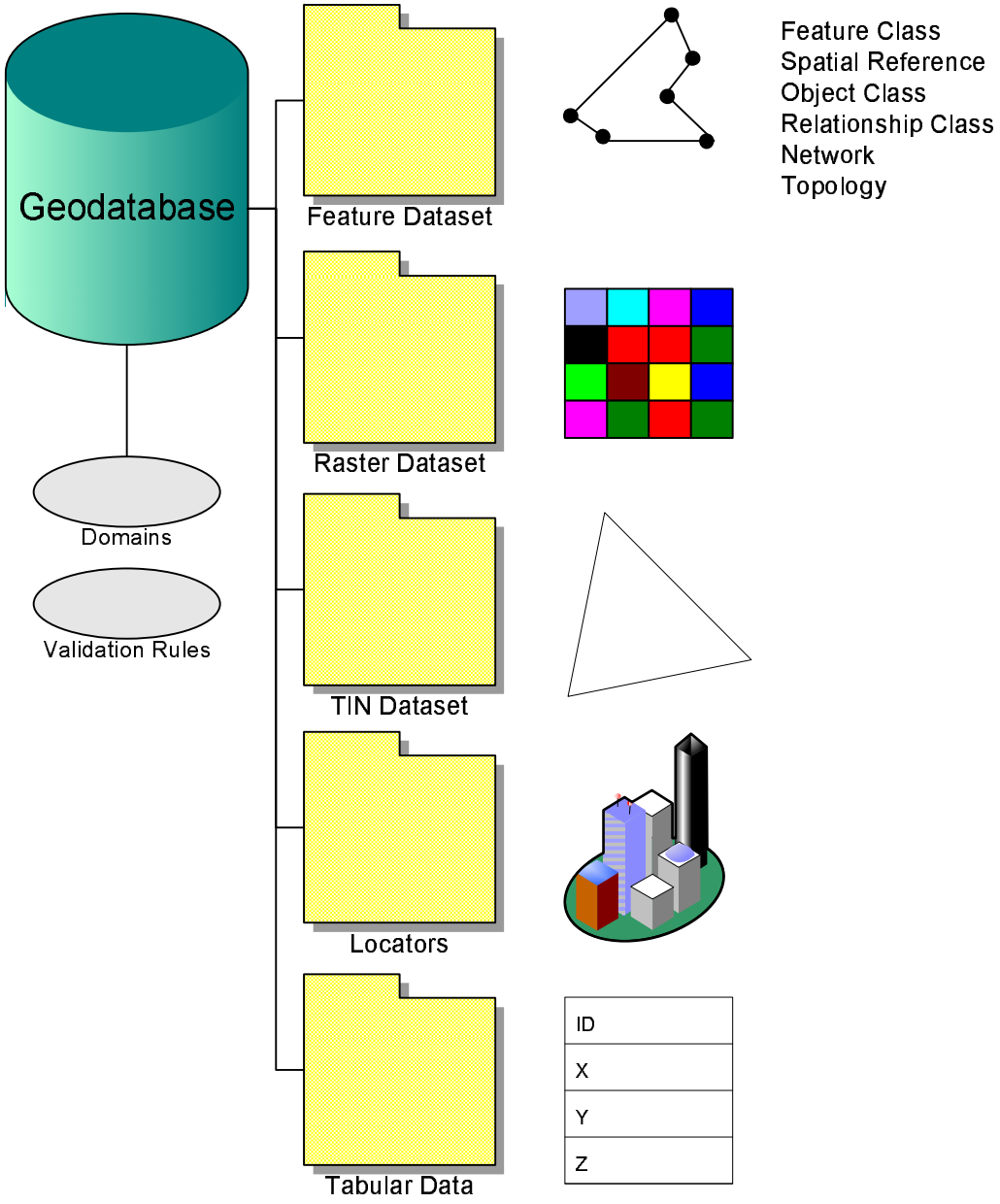



Figure 2.9: Geodatabase composition.



www.manaraa.com

database, but holds descriptive information instead. An object class could be created of land owners in a floodplain area, without storing any geographic data about a land owner's properties. Feature classes and object classes interact via a relationship class, which relates data in these classes or other tables (Zeiler, 1999).

Feature datasets also optionally store geometric networks and planar topologies. Networks are used when it is necessary to follow the path of a set of connected lines, such as on a stream or transportation network. Planar topologies define the behavior of polygon and line features sharing common boundaries, such as the boundary between a lake and its shoreline. When one of the entities changes, it forces a change in the other. For example, with planar topologies defined, editing a lake polygon to show the increased storage of an expanded reservoir would also update the surrounding polygons in the watershed.

Raster datasets are used to store raster information like images, aerial photographs, or a set of gridded elevation data used to define the surface of an area. The gridded elevation data can be used with automated tools to delineate watersheds.

TIN datasets are used to store triangulated irregular networks. TINs form surfaces that are also convenient for delineating watersheds, and are comprised of nodes, edges and faces.

Locators are used to store addresses and other locational information. An address could consist of geographic location such as an XY coordinate, an identification code, or a place name. This information is used for labeling objects or creating new features on a map. Linear referencing, used for making measurements of locations along a linear system, interpolates data along lines, such as for mile markers or distances from a starting reference. In the field of water resources, locators could be used for storing the locations of gages or stationing a river.

In addition, a geodatabase holds tabular data, domains and validation rules. Domains define valid values in attribute tables. Validation rules help to enforce other relationship rules, such as the connectivity between objects or features (Zeiler, 1999). For example, a validation rule can be created to ensure that each stream polyline is located inside a



www.manaraa.com

drainage area polygon, or that streams can only connect to other streams and not other kind of line features.

### 2.5.3 Geodatabase functionality

Geodatabases are used directly in a GIS environment through queries in order to extract certain information, or extended through special tools such as ArcObjects.

The user of GIS software can load a geodatabase, and view or edit both its spatial and attribute data. The GIS will be able to look at the relationship classes and behavior of the geodatabase to see how different parts of the data interact.

Both spatial queries and database queries can be performed on the geodatabase to extract useful information. Using the queries built into the GIS software or a database program, the data is easily accessed by those unfamiliar with the way the data is stored or how the GIS interface is used.

Another way a geodatabase is accessed is through ArcObjects. This is a set of GIS tools which are used by a computer programming language (such as Visual Basic, Delphi or C++) to access the geodatabase. ArcObjects allows a computer programmer to use any of the capabilities of the ArcGIS software to extract or manipulate data in the geodatabase (see Section 3).

## 2.6 FEMA Data Capture Standards

Geodatabases are an ideal tool to store digital FIRMs. In this form, the digital layers in the FIRM are stored in an easy to access geodatabase format, and are easily queried and related to other sets of data. In creating DFIRMs, FEMA has created a set of Data Capture Standards (DCS) for managing data between different mapping partners. The data standards define the types of data for terrain, survey, hydrology and hydraulic studies, as well as the minimum data sets required for floodplain maps (Federal Emergency Management Agency, 2004a).



www.manaraa.com

### 2.6.1 Terrain data

While just about any type of terrain data, such as printed USGS quad maps, can be used in a FEMA flood insurance study, digital data is preferred, because it is more easily used in future studies. Digital data also allows for the reproducibility of the results, as variations can occur when calculations are done manually.

According to FEMA, the best type of data to represent the terrain is a TIN, although DEMs are also acceptable and useful when putting together a hydrologic model. Table 2.3 (Federal Emergency Management Agency, 2004a) shows the accepted data formats for terrain data in a DFIRM project.

Table 2.3: Accepted terrain data formats for a DFIRM.

| **Format** | **References** |
|---|---|
| ArcInfo TIN Generate (line) | www.esri.com |
| ArcInfo TIN Generate (point) | www.esri.com |
| AutoCAD DXF version 12 | www.autodesk.com |
| ESRI Grid ASCII | www.esri.com |
| Comma delimited point text | |
| Space delimited point text | |
| LAS Binary LIDAR points | www.lasformat.org |
| 3-D Point shapefile | www.esri.com |
| 3-D Polyline shapefile | www.esri.com |

The information in a TIN represents the surface of the terrain indicating key breaks or changes in the elevation, as shown in Figure 2.10. Elevation contours are extracted from the TIN and stored in a CAD DXF file or polyline shapefile, while elevation values are stored



www.manaraa.com

either as points or as a grid with the remaining file formats. Gridded formats are also used to store hydrologically corrected DEMs or flow vectors, such as shown in Figure 2.11.

### 2.6.2 Survey data

The survey data collected for a DFIRM study includes cross sections, elevation reference marks, high water marks, and structures such as bridges, channels, culverts, dams and levees. These data, such as the cross section shown in Figure 2.12, are stored in a GIS as database tables, spatial tables, and photographs as necessary. The data collected from the surveys are used to help develop hydrologic and hydraulic models.

### 2.6.3 Hydrology data

The hydrology data collected to create the DFIRM includes database tables, spatial files, and the model input and output files of the hydrologic model. Typically, HEC-1 or HEC-HMS models are used for modeling. Table 2.4 shows the minimum required hydrologic data to be included in a DFIRM study (Federal Emergency Management Agency, 2004a), and typical formats of the data.

### 2.6.4 Hydraulic data

The hydraulic data collected to create the DFIRM includes database tables, spatial files, and the model input and output files of the hydraulic model. HEC-RAS or other models approved by FEMA (Federal Emergency Management Agency, 2004b) are used to generate water surface elevations. Table 2.5 shows the minimum required hydraulic data to be included in a DFIRM study (Federal Emergency Management Agency, 2004a), and typical formats of the data.



www.manaraa.com

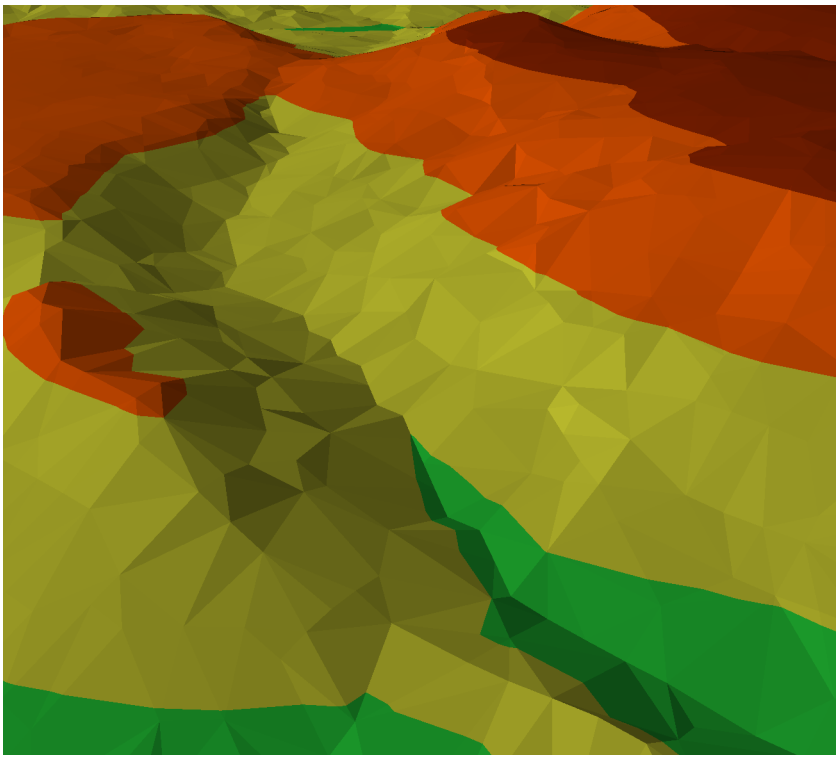


Figure 2.10: A TIN representing the terrain surface.

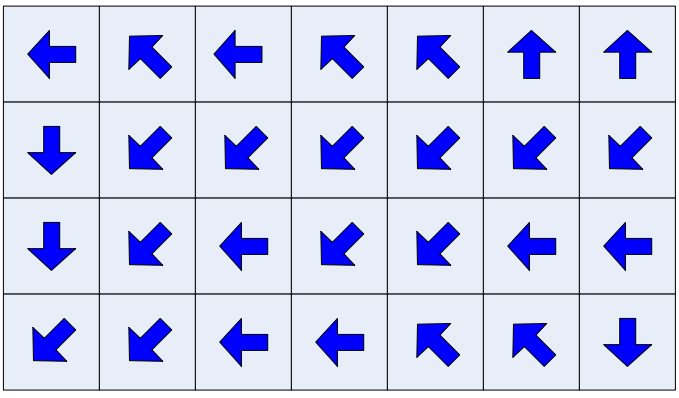
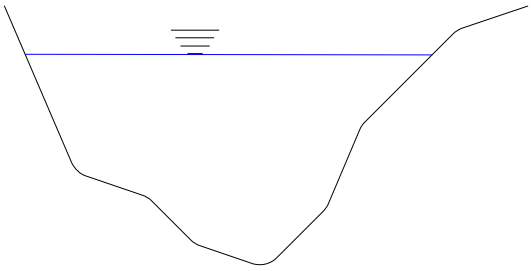


Figure 2.11: Flow vectors stored in a DEM.



www.manaraa.com



| CSCode | CrossM | Elevation |
|---|---|---|
| 0 | 0 | 75 |
| 0 | 30 | 66 |
| 0 | 60 | 65 |
| 0 | 80 | 62 |
| 0 | 110 | 61 |
| 0 | 140 | 64 |
| 0 | 150 | 69 |
| 0 | 190 | 74 |
| 0 | 220 | 75 |

Figure 2.12: A cross section as part of a DFIRM survey.

Table 2.4: Minimum required hydrology datasets.

| **Dataset** | **Format** |
|---|---|
| Stream channel network | polyline shapefile |
| Discharge points | point shapefile |
| Watershed areas | polygon shapefile |
| Optional geospatial datasets | shapefile |
| Table of discharges | database table |
| Model input/output files | (from the hydrologic model) |
| Summary of key data | database table |
| Report | |



www.manaraa.com

Table 2.5: Minimum required hydraulic datasets.

| **Dataset** | **Format** |
|---|---|
| Stream channel network | polyline shapefile |
| Cross sections (for 1-D model) | polyline shapefile |
| DTM file (for 2-D model) | DEM or other DTM format |
| Flood boundary (100, 500 year) | polyline shapefile |
| Model input/output files | (from the hydraulic model) |
| Model parameters (ex: n-value polygons) | shapefiles, other |
| Summary of key data | database table |
| Report | |



www.manaraa.com

www.manaraa.com

# 3 ArcObjects in Floodplain Mapping

Geodatabases can be used to store data required for analysis in water resources projects. By using the tools in ArcObjects, hydrologic and hydraulic information is set up by different engineers, government agencies and other groups in a standardized schema (such as the ArcHydro data model) for flood insurance studies and water resources modeling. ArcObjects are used to expand and customize the model as needed, not only from within ESRI's ArcView GIS, but also outside in other customized computer models.

## 3.1 GIS Integration

ArcObjects is the development platform for Environmental System Research Institute's ArcGIS Desktop suite, consisting of the software programs ArcInfo, ArcEditor, and ArcView. ArcObjects allows developers to take advantage of existing GIS functionality and expand on it as a separate application or integrated with existing software. The initial purpose of this research was to use ArcObjects inside of the Watershed Modeling System (WMS) software. WMS will be used as an example of using ArcObjects within existing water resources software to write out hydrologic, hydraulic and floodplain modeling data to a geodatabase.

### 3.1.1 Microsoft COM technology

ArcObjects routines are accessed through Component Object Model (COM). COM is a software standard for creating software components that are both programming language-independent and location-transparent (Gordon, 2000).

Components are binary pieces of software that are reusable between different computer programs and can be used as building blocks for a software application. COM provides



www.manaraa.com

standards for these components to allow them to communicate with one another, regardless of their source. Component objects are composed of one or more interfaces, which are used by the component object to expose its services or functionality.

COM objects can be written in many different computer programming languages, provided that they can access functions through pointers (Williams & Kindel, 1994). Some examples include Visual Basic, C#, and C++. They are then reused in other software programs using other programming languages because the COM standard works at the binary level. Figure 3.1 shows how an application would use COM to take advantage of different algorithms on different servers (either across a network or the COM server on the local computer) based on different programming languages.

COM also provides for location-transparency, which allows data access across networks between client and server computers, as well as only on a client computer. Components are accessed from the same process, the same computer, or another computer in a network, through interfaces in the component objects (Gordon, 2000).

### 3.1.2 Interfacing ArcGIS Desktop

ArcObjects use COM technology, which allows them to be accessed from other programs. This extends the GIS functions in ArcDesktop. New programs can be built from the ArcObjects components, or separate programs can use ArcObjects to tap into the capabilities of programs such as ArcView. Any programming language that supports COM may take advantage of ArcObjects. This research applies COM with C++.

Support for interfacing ArcObjects in C++ is provided through the Active Template Library. In C++, the `#import` command is used to access a type library and its associated files that come with ArcGIS Desktop, (see Appendix A). When the type library is compiled into the C++ program, each of the interfaces in ArcObjects are accessible (Zeiler, 2001).



www.manaraa.com

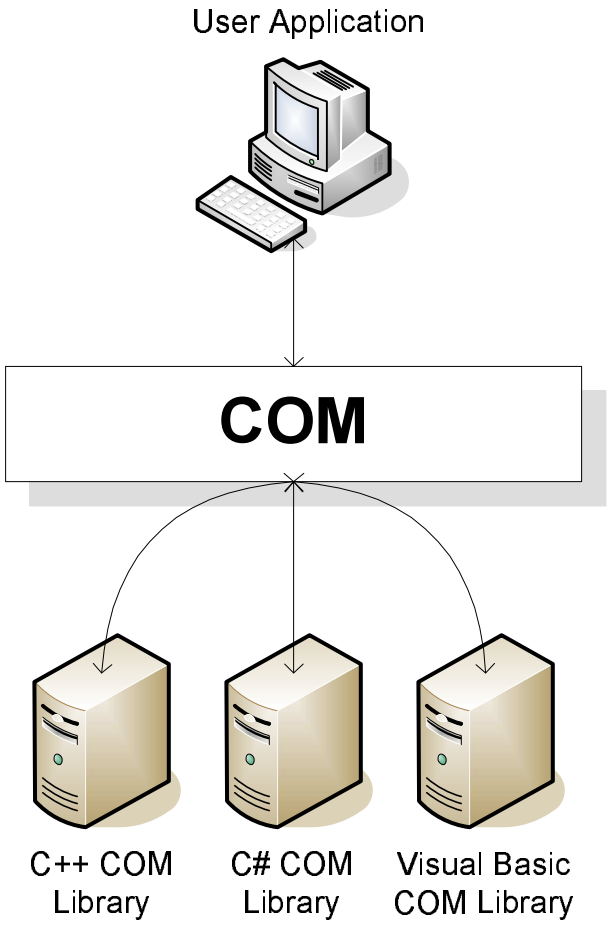



Figure 3.1: Using different programming languages together with COM.



www.manaraa.com

### 3.1.3 Extending and customizing a GIS

By accessing ArcObjects through a computer program totally separate from the ArcGIS Desktop, the capabilities of the GIS are used and customized according to the needs of the client program. The interfaces of the GIS components handle and process GIS data, while new interfaces are written to add new functionality to the GIS. For example, ArcObjects can be used to take floodplain mapping data outside of the GIS to build spatial maps within the GIS. This research was initiated to investigate the possibility of using ArcObjects in WMS in order to write out spatial and non-spatial data inside a geodatabase and other GIS formats.

## 3.2 Geodatabases for Water Resources

Flood maps and other hydrologic and hydraulic data are stored in geodatabases based on the ArcHydro data model. The model has been defined in such a way as to facilitate uniformity of data management in water resources problems by defining the relationships and behavior of the individual datasets, and is ideally suited for use as a FEMA DFIRM. The ArcHydro data model is flexible enough to be used by any program for storing hydrologic, hydraulic and flood data in a common format.

### 3.2.1 ArcHydro Data model

The ArcHydro data model was developed as a way of handling surface water hydrology and hydrography within a GIS. The model was developed through input from the GIS in Water Resources Consortium of the University of Texas at Austin. Formed by ESRI and the Center for Research in Water Resources (CRWR) at the University of Texas at Austin, the consortium took input from industry, government agencies and academia to develop the model. The current model was built and tested at CRWR (CRWR, 2004).

The ArcHydro data model incorporates both geospatial and temporal data in a GIS in order to describe the water resources of a given area (ESRI, 2002). The model includes



www.manaraa.com

such information as watersheds and drainage areas, river channels, elevation data, gage stations, topography, and precipitation. The data are stored in the GIS as a geodatabase consisting of various feature datasets, feature classes, tables and relationship classes. The main components of the ArcHydro data model are shown in Figure 3.2 and include the Network, Drainage, Channel, and Hydrography feature datasets (Maidment, 2002).

The Network feature dataset in the ArcHydro data model forms the basis for the geodatabase. The Network feature dataset consists of the HydroEdge, HydroJunction, HydroNetwork_Junction, SchematicLink, and SchematicNode feature classes. These describe the topological relationship of the model, describing how different features in the model are connected and how water travels throughout the system. These feature classes are used to establish junctions in river networks and streams, flow directions, boundaries between water bodies and land, as well as connections between rivers, streams and lakes. In addition to connections to water bodies, junctions are also used to make connections with feature classes in other feature datasets such as watersheds and monitoring points. Figure 3.3 shows a typical network feature dataset.

The Drainage feature dataset describes the drainage pattern of the area of interest. The ArcHydro data model classifies drainage areas as Basins, Watersheds, or Catchments, while using the DrainagePoint feature class as outlets for the drainage areas and the DrainageLine feature class for drainage paths. Watersheds are used as subdivisions of basins, while catchments are used as subdivisions of watersheds. Drainage areas, paths and concentrations points are determined from elevation data, by tracing the theoretical elevation grid cell-to-cell flow of a drop of water at each elevation point (Peucker & Douglas, 1975; Jenson & Domingue, 1988). A set of watersheds consisting of streams (DrainageLines) and outlets (DrainagePoints) are shown in Figure 3.4.

River channels are modeled in the geodatabase in the Channel feature dataset. The CrossSection and ProfileLine feature classes are used to define the area of flow and changes in elevation along a river. These feature classes are important in defining river features



www.manaraa.com

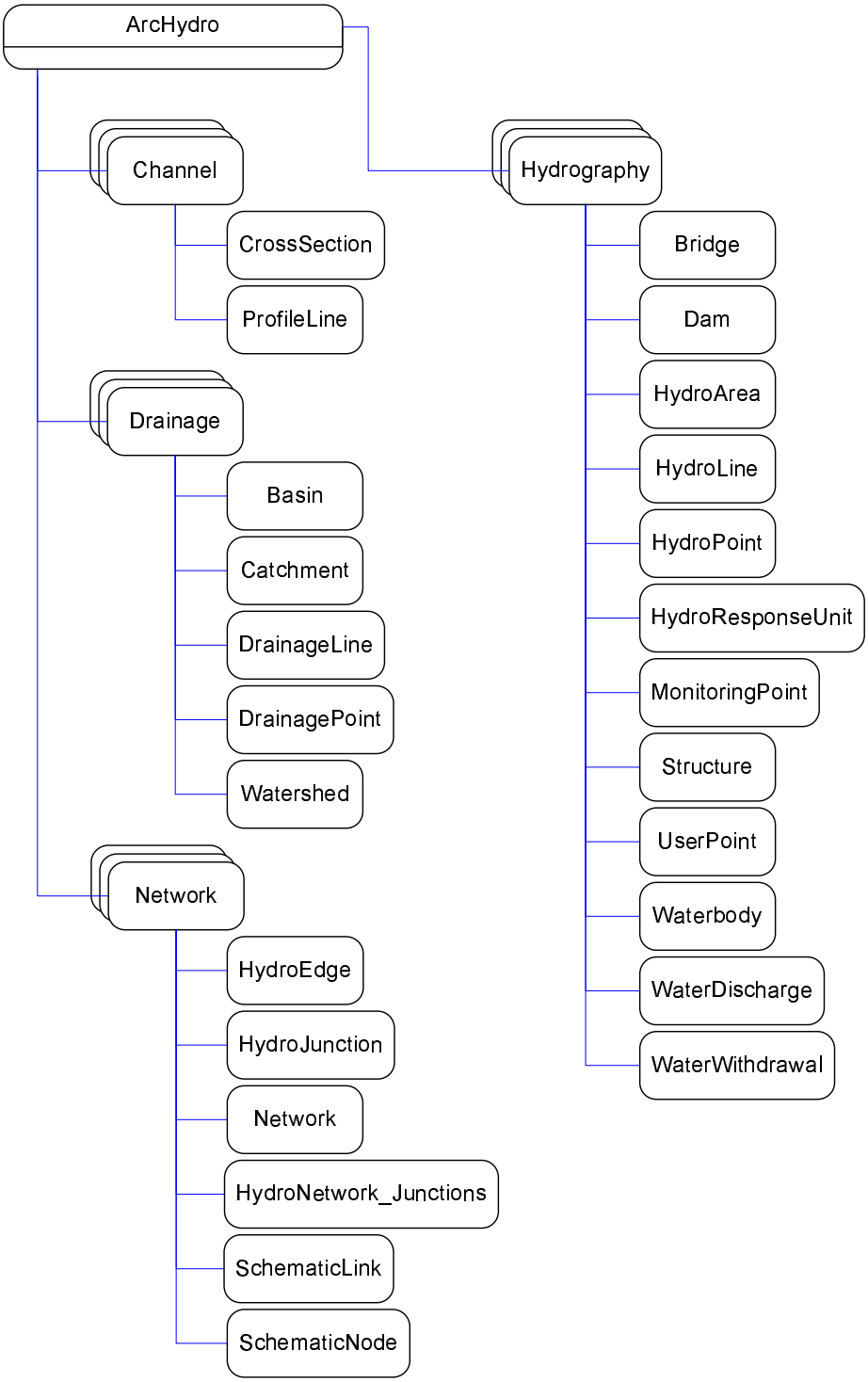



Figure 3.2: The ArcHydro feature datasets and feature classes.



www.manaraa.com

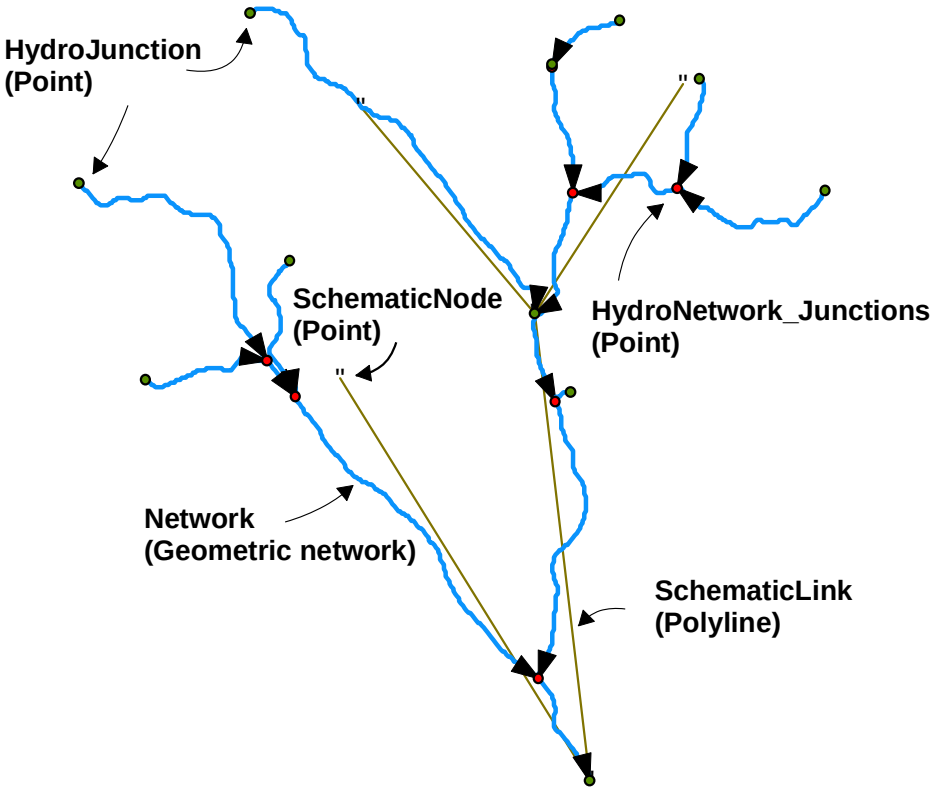



Figure 3.3: ArcHydro network dataset.



www.manaraa.com

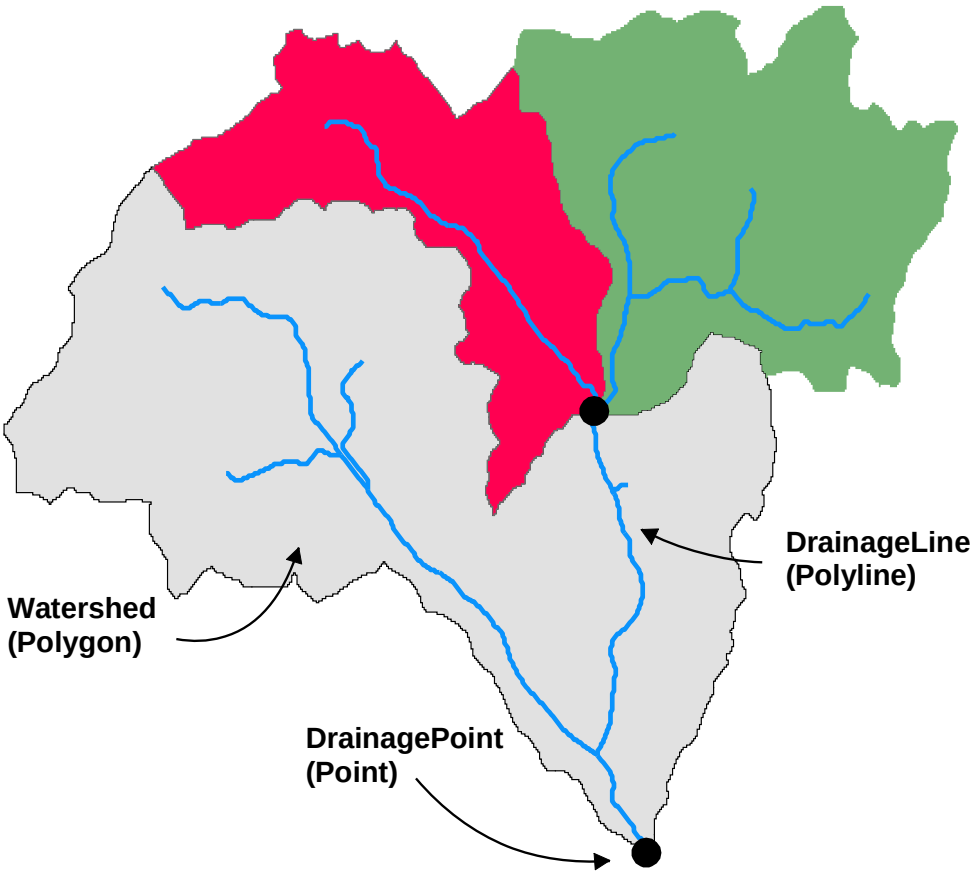



Figure 3.4: ArcHydro drainage feature dataset.



www.manaraa.com

such as the thalweg, banks, or delineating a floodplain. Figure 3.5 shows a set of cross sections along with channel and bank polylines.

## 3.3 Programmatically Creating a Geodatabase

Using ArcObjects, computer programs can be written to create an ArcHydro geodatabase outside of ArcView, and expand on it as necessary. The ArcObjects tools access the core GIS functionality to store data within the feature classes of the geodatabase.

### 3.3.1 Creating a Workspace

The first step in creating a geodatabase from scratch involves creating a Workspace Factory. A Workspace Factory allows for connecting to or even creating Workspaces, which represent a database. The database format typically used in the ArcHydro geodatabase is the Microsoft Access format, although other enterprise database formats also exist on high-end servers. Once the Workspace Factory is created, it is used to generate a new Microsoft Access .mdb file.

### 3.3.2 Creating feature datasets

The next step in creating the geodatabase involves creating the feature datasets for holding the relational data of the model. Each feature dataset should be defined by a particular spatial reference, which defines the geographic and projected coordinate system of the spatial data comprising the feature classes within the feature dataset. As defined by ESRI, each feature dataset has only one spatial reference at a time, and each of the feature classes within the feature datasets must have the same spatial reference. Individual feature classes containing collections of vector based feature objects can then be added to the feature datasets.



www.manaraa.com

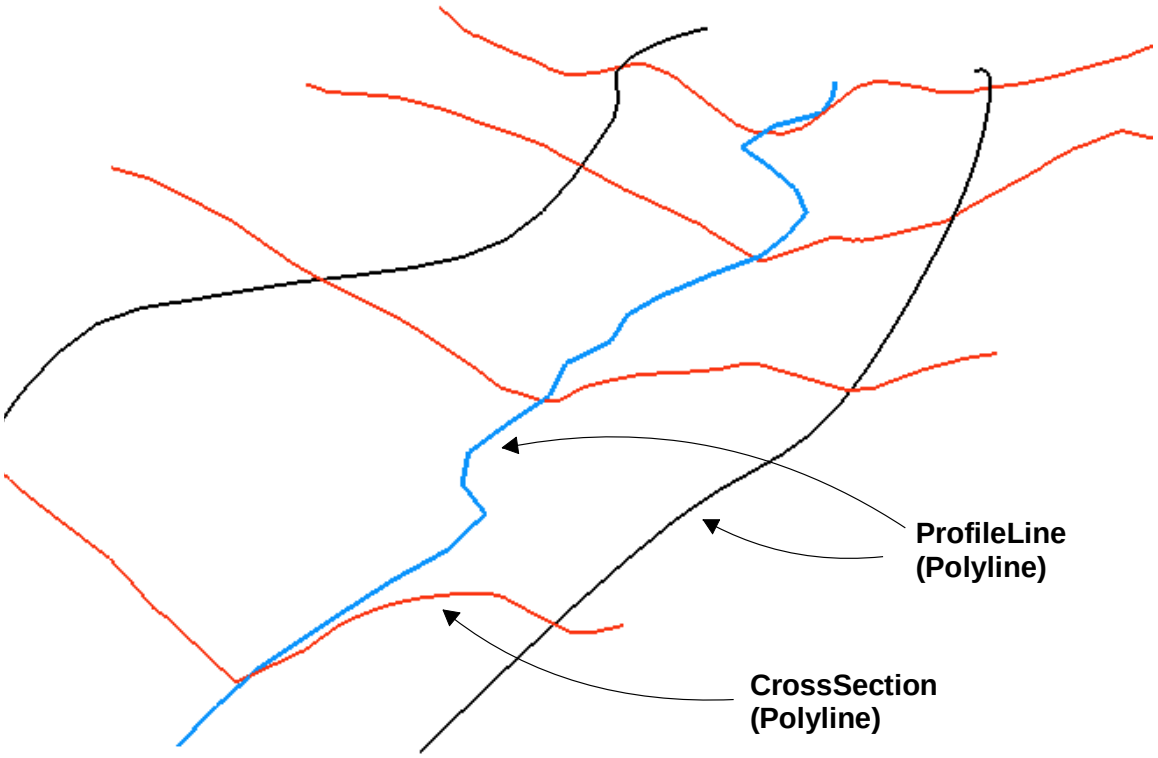



Figure 3.5: ArcHydro channel feature dataset.



www.manaraa.com

### 3.3.3 Storing features

While the feature objects are being added to the feature datasets, both geographic and tabular data can be stored, or the tabular data can be added later. At a minimum, fields containing an object ID and the geometry are required. The object ID is a unique value for each of the feature objects stored in the feature class.

A Feature must first be created and its Shape defined before it is added to the feature class. In order to create a point shape, its XY position must be known. In order to create a polyline shape, the XY position of each endpoint (nodes) and any internal points (vertices) comprising the polyline must be known. The same holds true for a polygon shape, although a single polygon could consist of multiple ring segments or contain holes.

Once the geometry has been added, the geodatabase can be extended to include other important relational data. In the ArcHydro database, a HydroID field is used to uniquely identify a particular feature. Other fields are used to make relationships with other features or to store important information needed to set up a hydrologic or hydraulic model. For example, the HydroJunctionHasWatershed relationship class creates a one to many relationship between the HydroID of a HydroJunction feature class with the JunctionID of the Watershed feature class as shown in Figure 3.6.

### 3.3.4 Data sources for ArcHydro feature classes

The basic ArcHydro data model consists of four feature datasets, each consisting of various feature classes (see Figure 3.2) (Maidment, 2002). WMS is used to develop data sources for many of these feature classes, provided vector based data is being used for hydrologic and hydraulic analysis, or if other forms of data are converted to vector based data. These data are stored in various groupings called coverages. The drainage coverage shown in Figure 3.7, consists of basin polygons, stream arcs, and outlets points. These data set up the Drainage feature dataset in the geodatabase. Table 3.1 shows the relationship between ArcHydro feature classes and data created by WMS to write out an ArcHydro geodatabase.



www.manaraa.com

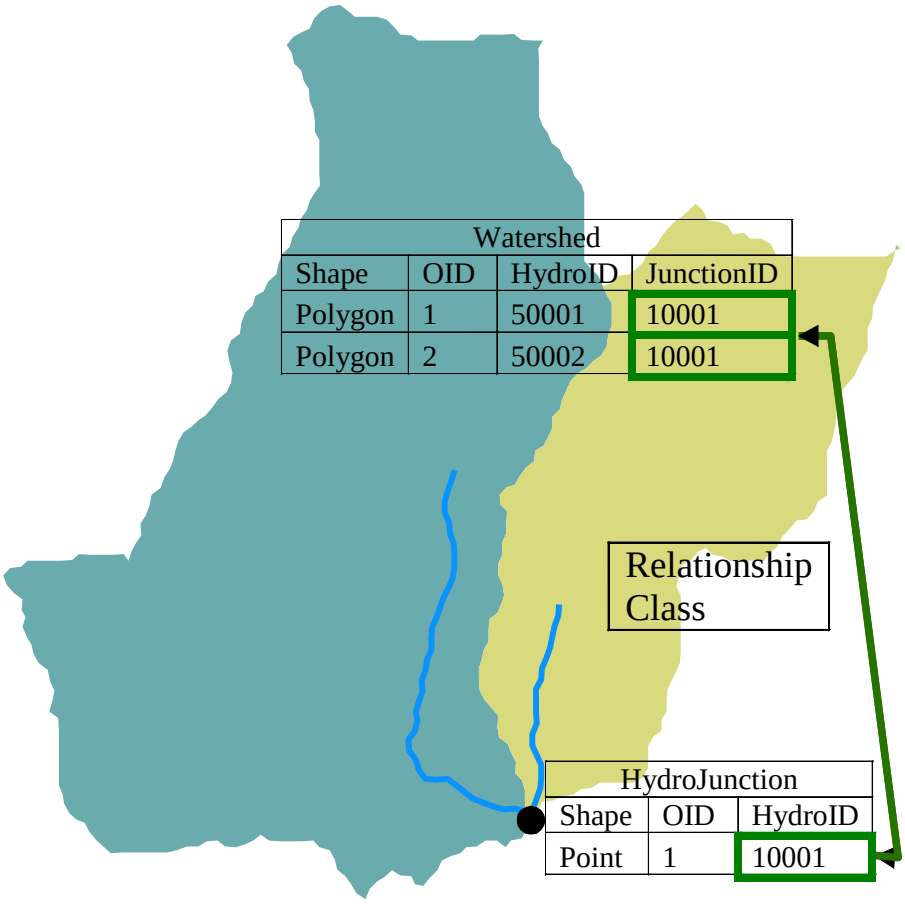



Figure 3.6: Relationship class between feature classes.



www.manaraa.com

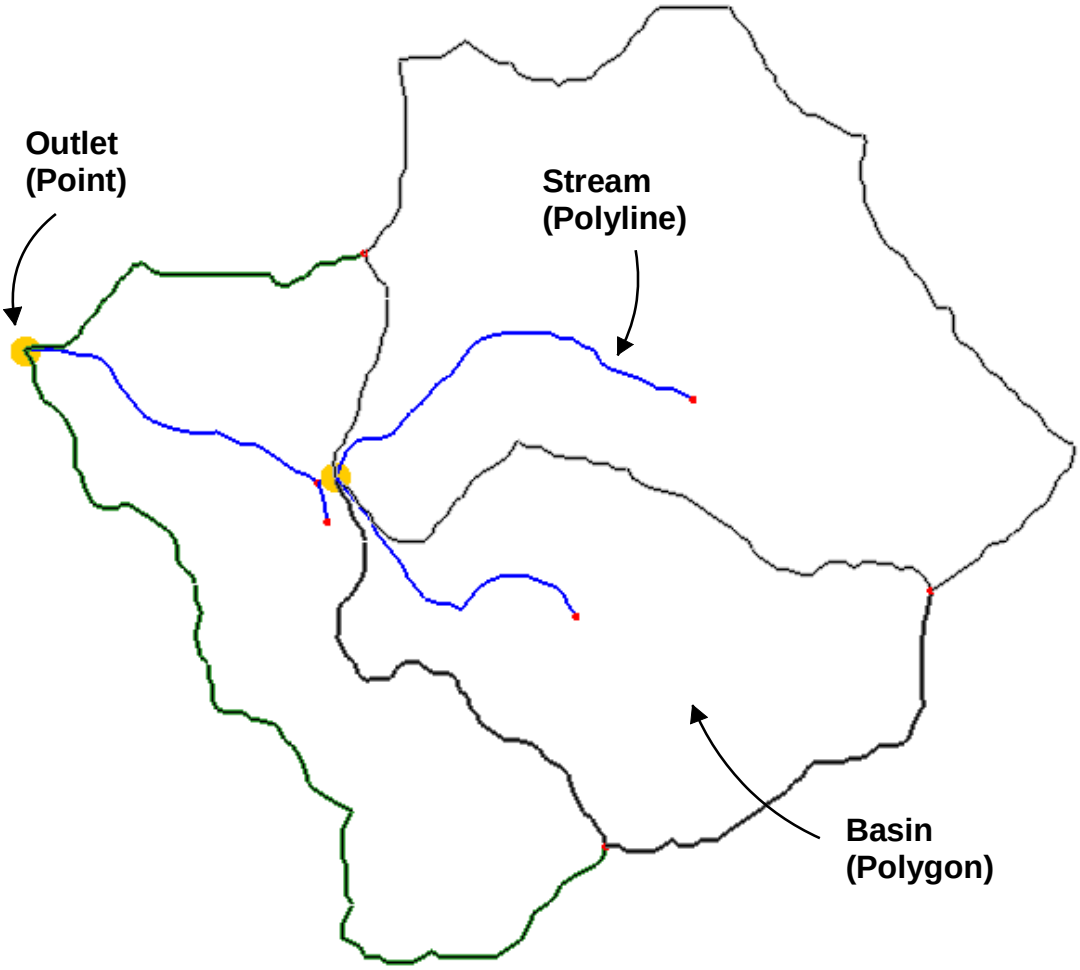



Figure 3.7: Vector data comprising a coverage in WMS.



www.manaraa.com

Table 3.1: WMS data sources for ArcHydro feature classes

| **ArcHydro** | | **WMS** | |
|---|---|---|---|
| Feature dataset | Feature class | Coverage | Feature type |
| Network | HydroEdge | Drainage | Arcs |
| | HydroJunction | Drainage | Points |
| | HydroNetwork_Junction | Drainage | Points |
| | SchematicNode | Drainage | Points |
| | SchematicLink | Drainage | Arcs |
| Drainage | Basin | Drainage | Polygons |
| | Watershed | Drainage | Polygons |
| | Catchment | Drainage | Polygons |
| | DrainagePoint | Drainage | Points |
| | DrainageLine | Drainage | Arcs |
| Channel | CrossSection | 1D Cross Section | Arcs |
| | ProfileLine | 1D Centerline | Arcs |



www.manaraa.com

Much of the data stored in WMS in a drainage coverage is duplicated throughout the ArcHydro database, as seen in Table 3.1. For example, the stream arcs stored in the drainage coverage are used to define the DrainageLine feature class in the Drainage feature dataset, as well as the HydroEdge feature class and the Network of the Network feature dataset. Outlet points in the WMS drainage coverage are also used to define the HydroJunction and DrainagePoint feature classes in the Network and Drainage feature datasets in the ArcHydro data model.

The CrossSection feature dataset is built from two coverages in WMS, the 1D hydraulic centerline and 1D cross section coverages. The first coverage defines the river or stream channel along with its banks, while the cross section coverage defines cross sections with the help of a database. These coverages in WMS, shown in Figure 3.8, relate directly to the ProfileLine and CrossSection feature classes in the ArcHydro data model.

The Network dataset is built from the drainage coverage in WMS, along with the hydrologic modeling tree. The drainage coverage is used to define the Network, HydroJunction and HydroEdge feature classes, while the hydrologic modeling tree serves as the basis for the SchematicLink and SchematicNode feature classes. Figure 3.9 shows a drainage coverage in WMS and a hydrologic modeling tree.

## 3.4 Creating a Geodatabase With WMS

Hydrologic and hydraulic data can be taken from WMS or other GIS programs and placed in an ArcHydro geodatabase using ArcObjects and the C++ programming language. The thesis only considers using WMS to create an ArcHydro geodatabase, though the methods and source code shown are applicable to other programs. In addition, the ArcHydro geodatabase can be expanded to hold other important information such as the floodplain and flood probability maps. Refer to Appendix A for C++ code samples.



www.manaraa.com

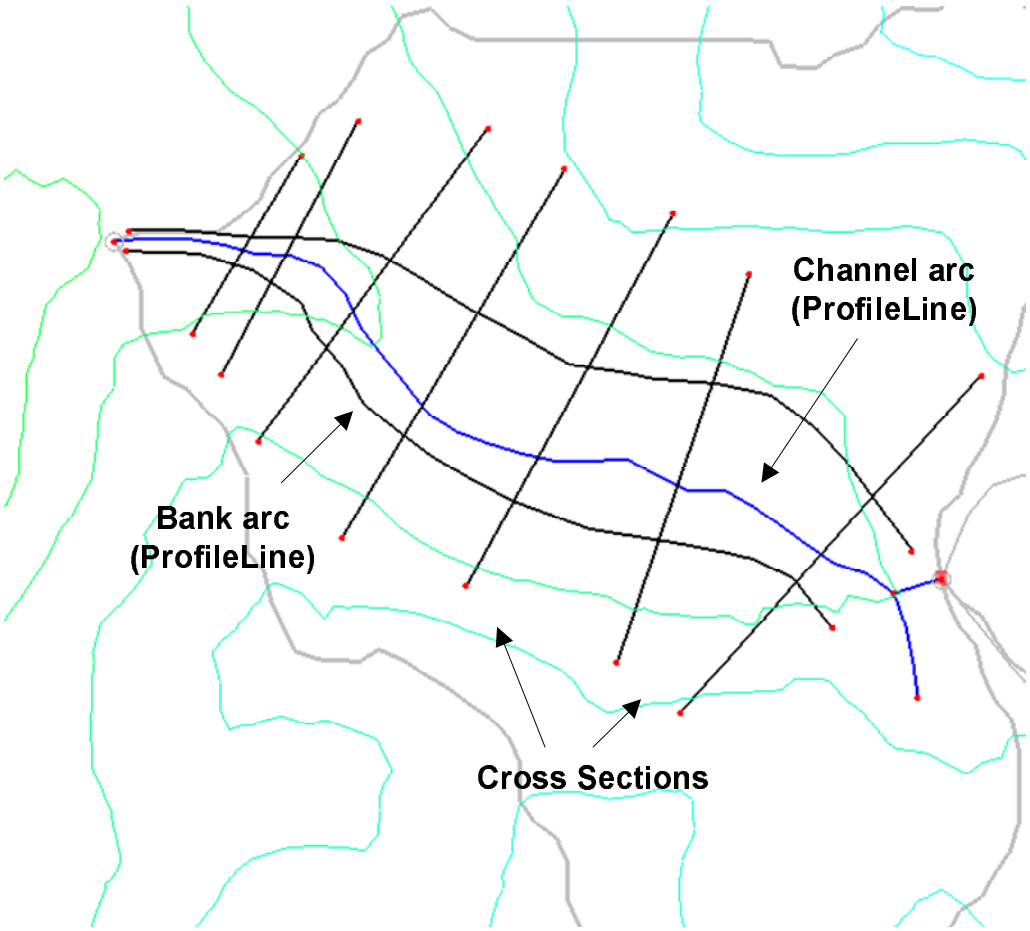



Figure 3.8: Hydraulic centerline and cross section coverages in WMS.



www.manaraa.com

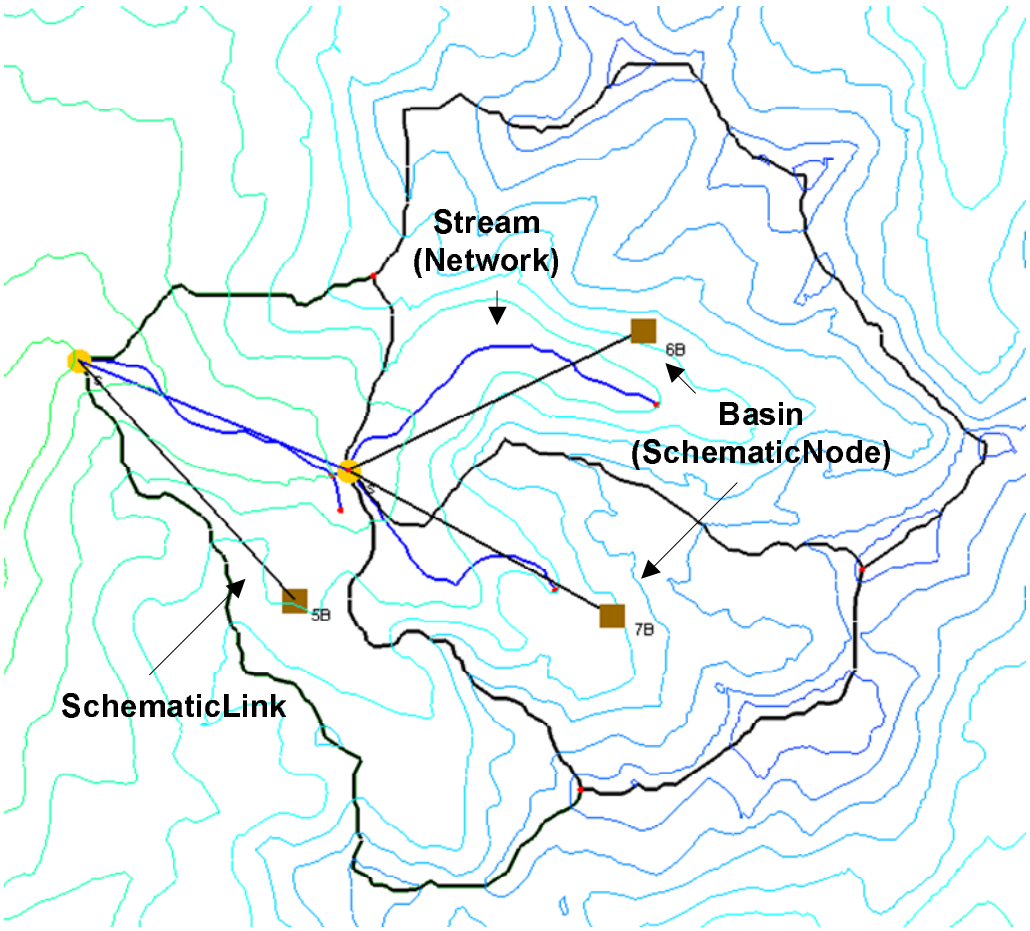



Figure 3.9: WMS drainage coverage and hydrologic modeling tree.



www.manaraa.com

### 3.4.1 Geodatabase preparation

In order to create a geodatabase for floodplain mapping within WMS, the appropriate data must first be collected and hydrologic, hydraulic and floodplain models run. In WMS, the data are usually in the form of digital elevation models or triangular irregular networks for the elevation, along with vector data called feature objects in order to store and calculate the parameters required by the models.

### 3.4.2 Create a workspace

The first step is to create a workspace factory, from which point a geodatabase workspace can be formed (in Microsoft Access format). Figure A.5 shows how a workspace factory is created using C++.

### 3.4.3 Define a spatial reference

The next step involves defining a spatial reference for the geodatabase. Each individual feature dataset inside a geodatabase must have its own, common spatial reference defined. Inside WMS, the coordinate system set for the current project is used to define the spatial reference, whether it is a geographic coordinate system, a Universal Transverse Mercator projected coordinate system, or a state plane coordinate system. A spatial reference factory is instantiated, at which point it is used to create a projected or geographic coordinate system. The spatial reference's properties are set to the new coordinate system. The code in Figure A.2 shows how to create a geographic coordinate system, with a datum such as the NAD 1983 datum, one of many international coordinate systems supported by ArcObjects. Projected coordinate systems are also created, as shown in Figure A.3.

### 3.4.4 Create ArcHydro datasets

Once a spatial reference has been created and defined, it is used to help set up a feature dataset. To create the feature dataset, a feature workspace is first created from the instance



www.manaraa.com

of the Access workspace. The creation of a feature workspace and a feature dataset are shown in Figures A.5 to A.9.

Different datasets should be created for each of the standard datasets in the ArcHydro data model. These datasets are then used to store the drainage coverage from WMS, consisting of vector data defining the basins, stream networks and outlet points, and the 1D hydraulic coverages defining the cross sectional data.

### 3.4.5 Storing feature classes in feature datasets

For each dataset created, individual feature classes are then formed based on the vector data in WMS. Each feature class stores similarly grouped features with common attributes. In order to create a feature class, a set of fields for these attributes are created, including object ID (OID) and geometry fields. Once these two necessary fields are defined, other fields are added, such as those required by the ArcHydro data model. An example of how to create the required OID and Shape fields is shown in Figure A.10. Additional fields are then added for each feature class, as demonstrated in Figure A.11.

With the fields defined, a feature class is created inside of a feature dataset. Feature classes should be defined based on the ArcHydro data model (Maidment, 2002). Most of the feature classes will be comprised of simple features (points, polylines and polygons), while some of the feature classes found in the Network feature dataset will be comprised of complex edges or simple junctions. Figure A.12 shows the method of creating a simple feature class, given a set of previously defined fields. The fields contain the geometry definition, such as what types of features will be stored in the class (points, polylines or polygons).

Once a feature class has been created, the individual point, polyline and polygon features are created and stored in the class. Vector based feature objects in WMS are used as the basis for these features. The process involves creating a new feature for the feature class, defining its shape or geometry information, adding information into the table for the new



www.manaraa.com

feature, and storing the feature in the feature class. This process is demonstrated in Figure A.13, where the point's X and Y positions are set, along with a value in its HydroID field. For example, when using the outlet point data in WMS to create the DrainagePoint feature class, a new point feature is created for each outlet point. The XY position of an outlet point is used to set shape field of a new feature. Information from the outlet point, such as its ID and name are then used to fill in the remaining fields of the new feature, such as the HydroID and HydroCode (the permanent public identifier of the feature). Figures A.14 and A.15 show how to create polyline and polygon features.

### 3.4.6 Adding flood datasets to the geodatabase

An important part of this research was to store digital flood maps as part of a geodatabase. As discussed in Section 2, the result of the floodplain delineation model is a set of water surface elevations throughout the DTM. While these water surface elevation values could be used to create a new point feature class covering the DTM, they are more useful in the form of a polyline feature class in a new floodplain dataset. This new dataset is used to store a flood extent map, a flood probability map, and an AEP map, as outlined in Figure 3.10. While this flood dataset has not been accepted by a committee and is not part of the ArcHydro data model, this is the model I have used to implement the storage of digital flood maps generated by WMS into a geodatabase.

The flood extent feature class is developed by tracing stage values on the DTM that are equal in elevation to the DTM. By connecting these points on the DTM and interpolating between them, polylines are created showing the extent of the flooding. After creating a new flood extent feature class, these polylines are added according to the method in A.14. In addition to the OID and Shape fields, it is proposed that the following two fields be added to the features in the feature class:

- HydroID of the flood extent polyline
- Recurrence interval of the flood



www.manaraa.com

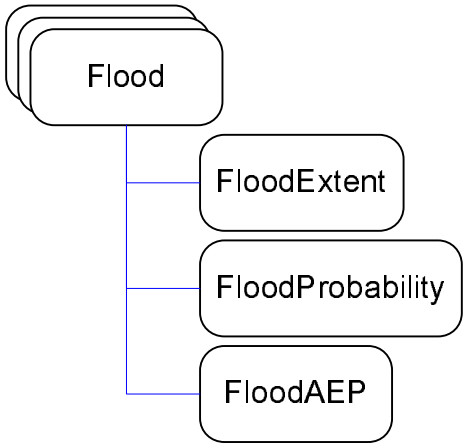



Figure 3.10: Flood dataset.

The HydroID attribute is a unique identifier for the flood extent polyline inside the ArcHydro geodatabase. The recurrence interval field identifies the flood recurrence interval used in the floodplain model to develop the flood extent map.

The flood probability map shows the probability of flooding at a single given return period. This map is usually displayed as a set of contours, such as shown in Figure 3.11. Contours intervals are then specified at regular intervals, to show the probability of flooding at levels such as 25%, 50%, 75%, or 100%. To add a flood probability map to the flood dataset, a new FloodProbability polyline feature class is created with the following fields:

- HydroID of the flood probability polyline
- Recurrence interval of the flood
- Flood probability value of the polyline

The HydroID uniquely identifies the polyline in the geodatabase. The recurrence interval of the flood is used to group similar polylines, in the case where multiple floodplain models are run for different return periods. The flood probability value stores the chance of flooding along the line. Figure 3.12 shows a FloodProbability polyline feature class created from flood probability contours.



www.manaraa.com

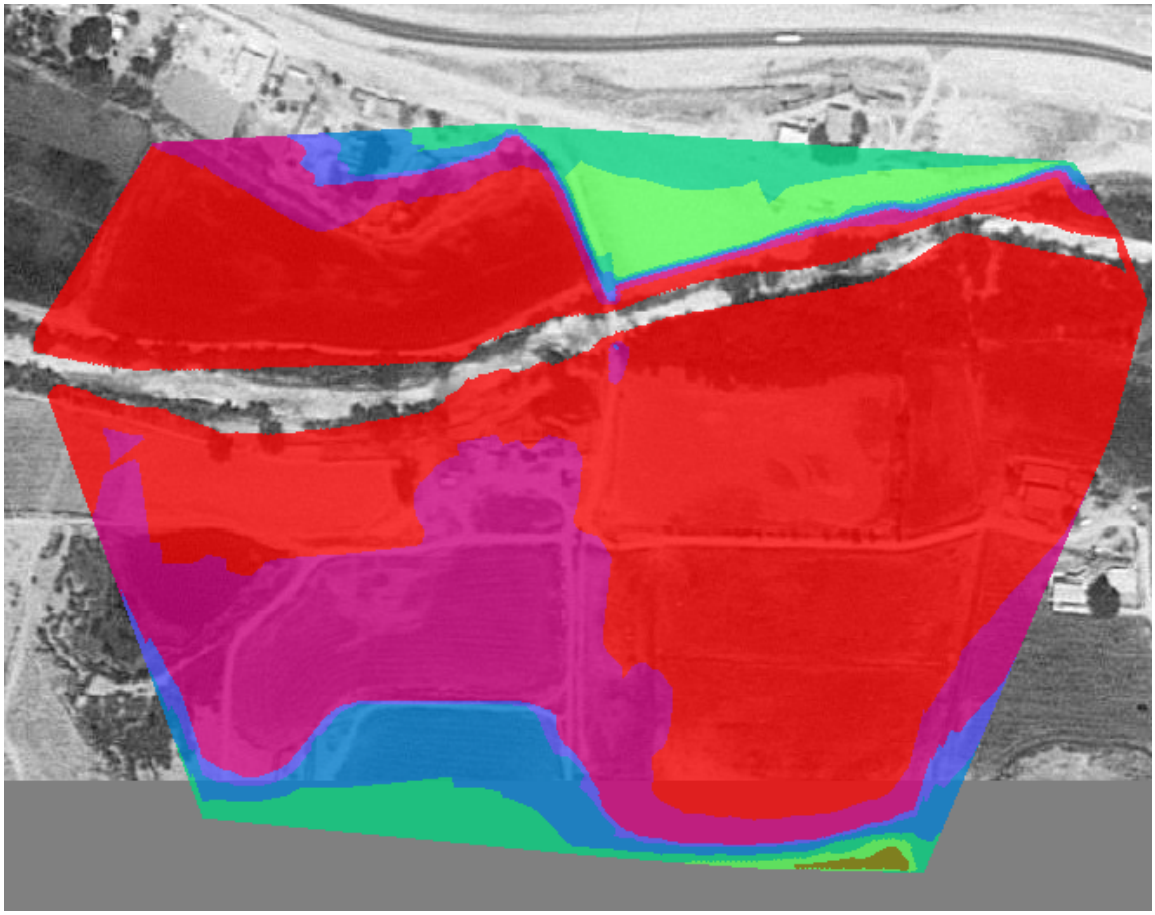


Figure 3.11: Flood probability contours.



www.manaraa.com

Figure 3.12: Polylines from flood probability contours.



www.manaraa.com

The AEP map shows the probability of flooding at each point on the DTM for any given year. Unlike the flood probability map, the AEP takes into account the probability of flooding at multiple recurrence intervals. Any recurrence interval floodplain can be determined from the AEP map through the equation $probability = (1/return\ period)$. Commonly used flood return periods and their associated probability of flooding are shown in Table 3.2, with the resulting AEP map shown in Figure 3.13. To add an AEP map to the flood dataset in the geodatabase, a new FloodAEP polyline feature class is created with the following fields:

- HydroID of the AEP polyline
- The percent probability of flooding along the polyline
- The associated return period of flooding

The HydroID of the polyline is used to uniquely identify the polyline in the geodatabase. The probability and return period values store the probability of flooding for any given year along the polyline. Polylines created from an AEP map are shown in Figure 3.14.

Table 3.2: Flood probabilities.

| **Return period (years)** | **Probability of flooding (%)** |
|---|---|
| 2 | 50.0 |
| 5 | 20.0 |
| 10 | 10.0 |
| 20 | 5.0 |
| 50 | 2.0 |
| 100 | 1.0 |
| 500 | 0.2 |



www.manaraa.com

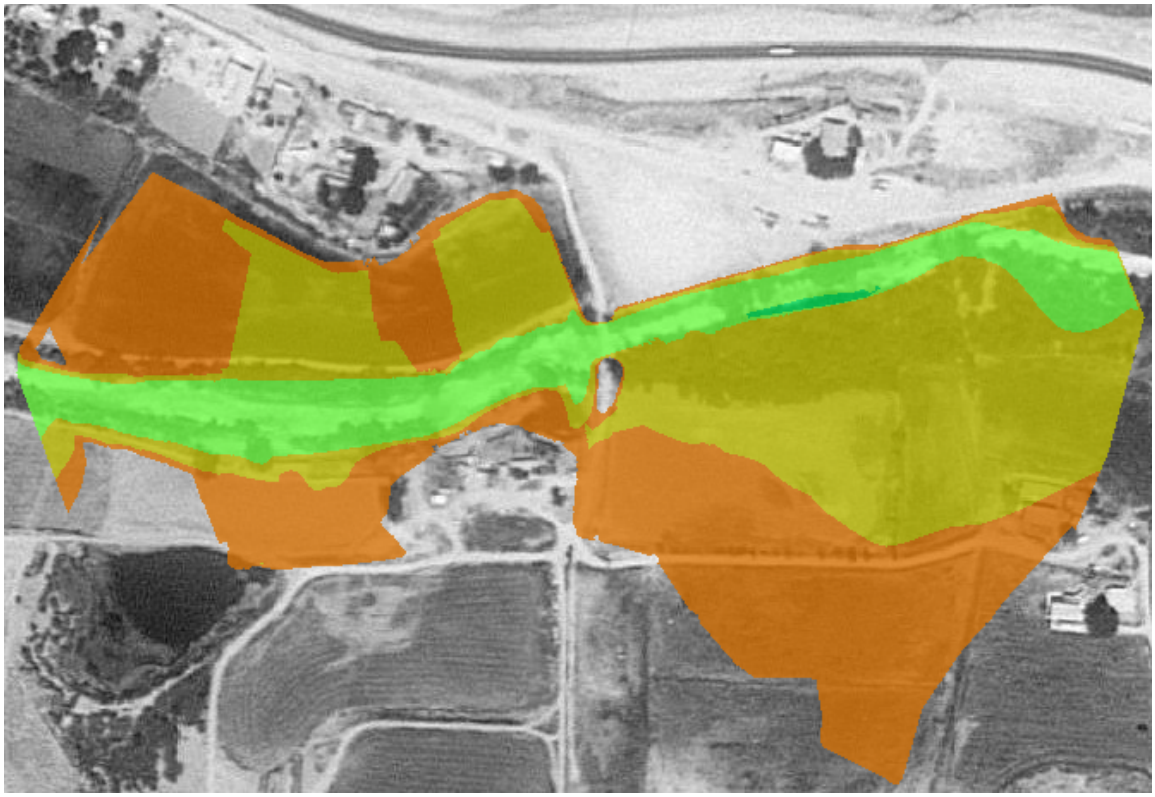


Figure 3.13: AEP contours.



www.manaraa.com

Figure 3.14: Polylines created from and AEP map.



www.manaraa.com

# 4 Conclusions

This research described using ArcObjects in a stand alone computer program and included methods for creating geodatabases using the ArcHydro data format and storing flood maps and flood probability maps as part of a geodatabase for creating digital flood insurance rate maps. ESRI currently has described a data model for storing watershed modeling data in a database, but only from within their GIS software. FEMA has set guidelines for updating and creating new flood hazard maps in digital format, though a variety of data standards can be used.

## 4.1 Technical Contributions

Several important advances in watershed and floodplain modeling data management were made through this research. The most important advances are as follows:

1. Using ArcObjects in stand alone engineering software in order to store hydrologic, hydraulic and flood mapping data in a common, easy to share data format. ArcObjects can be embedded into any program in order to aid in water resources modeling.

2. A method of managing and storing spatial, drainage, network and cross section datasets in an ArcHydro geodatabase from outside a GIS. ArcObjects were implemented in the program WMS as a way to store this data and to follow the ArcHydro geodatabase model. The same methods used in WMS can be used by other programs.

3. A method of storing flood maps and flood probability maps in a geodatabase. The ArcHydro geodatabase model did not define a flood dataset for storing the results of a floodplain delineation or uncertainty analysis. A flood dataset was created in



www.manaraa.com

order to produce flood maps meeting the FEMA digital flood insurance rate map specifications.

These methods and procedures make it possible to use specialized hydrologic, hydraulic and flood modeling software to create geodatabases for use in a GIS, such as shown in Figure 4.1. Flood maps can also be used to create digital flood hazard or digital flood insurance rate maps.

## 4.2 Future Research

The research presented in this thesis can be extended in many ways. These include:

1. Adding relationship classes to the flood feature dataset.
2. Expanding the tables in the flood feature dataset.
3. Extending the geodatabase through additional feature datasets.
4. Developing DFIRMs.

First, additional relationship classes could be developed in the geodatabase to better integrate the flood feature dataset with the ArcHydro data model. Existing relationship classes in the ArcHydro data model are used to link features in the Network dataset with other features in the Drainage and Hydrography datasets. Relationship classes could be defined between HydroJunctions and flood maps, and between CrossSection or ProfileLine features and flood maps. This would make the hydraulic data used to create the flood maps visible to the flood maps.

Second, additional fields and database tables could be developed to better describe the flood datasets. The fields in the flood datasets could be expanded to show information about flood stages or the methods used to create the flood probability maps. Database tables could be added to show the results at each iteration of the stochastic modeling used to create flood probability maps.



www.manaraa.com

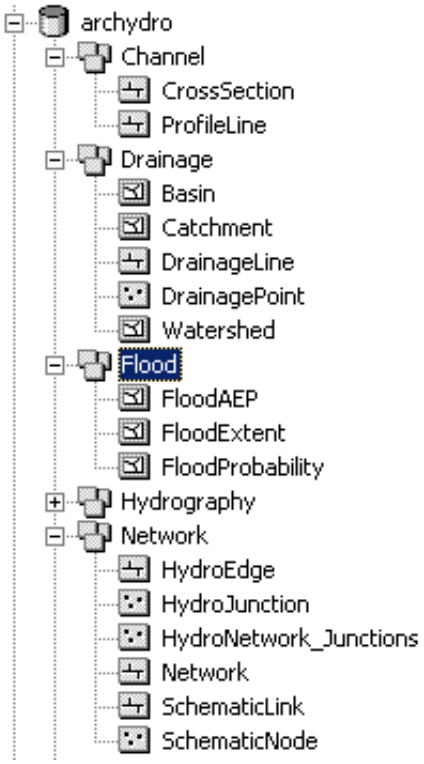



Figure 4.1: A geodatabase created in WMS shown in ArcCatalog.



www.manaraa.com

Third, other datasets could be developed for other areas in water resources, such as groundwater. While the HydroResponseUnit feature class in the Hydrography feature dataset in the ArcHydro data model is used to define areas of surface and sub-surface interaction, its use in groundwater modeling is very limited. No fields in the HydroResponseUnit feature class have been defined for groundwater modeling. A groundwater extension to the ArcHydro data model could be developed and added to the geodatabase.

Finally, improvements could be made to the flood mapping process to develop FEMA DFIRMS. The process of making a DFIRM could be made easier by further customizing the Flood feature dataset. Further integration between FEMA's DFIRM database and the geodatabase could be made, such as adding further attribute tables as required by FEMA.



www.manaraa.com

## References


Bedient, P. B. & Huber, W. C. (1998). *Hydrology and Floodplain Analysis*. Addison-Wesley.

CRWR (2004). GIS in Water Resources Consortium. http://www.crwr.utexas.edu/giswr.

Environmental Modeling Research Laboratory (2003). *Watershed Modeling System help file, WMS 7.1*. Environmental Modeling Research Laboratory.

ESRI (2002). *Online Help for ArcHydro Tools*. ESRI.

Federal Emergency Management Agency (2003). *Guidelines and Specifications for Flood Hazard Mapping Partners*. FEMA.

Federal Emergency Management Agency (2004a). *Guidelines and Specifications for Flood Hazard Mapping Partners*. FEMA.

Federal Emergency Management Agency (2004b). Numerical models meeting the minimum requirement of the NFIP. http://www.fema.gov/fhm/en_modl.shtm.

Federal Emergency Management Agency Map Service Center (2004). Digital Flood Insurance Rate Map, Colusa County, CA. http://msc.fema.gov/dfirm.shtml.

Gordon, A. (2000). *The COM and COM+ Programming Primer*. Prentice Hall PTR.

Jenson, S. K. & Domingue, J. O. (1988). Extracting topographic structure from digital elevation data for geographic information systems analysis. *Photogrammetric Engineering and Remote Sensing*, *54* (11), 1593–1600.

Maidment, D. R. (2002). *ArcHydro: GIS for Water Resources*. ESRI Press.




www.manaraa.com

Noman, N. S. (2001). *An integrated approach for automated floodplain delineation from digital terrain models*. Ph.d. diss., Brigham Young University.

Peucker, T. K. & Douglas, D. H. (1975). Detection of surface specific points by local parallel processing of discrete terrain elevation data. *Computer Graphics and Image Processing*, *4*, 375–387.

Smemoe, C. M. (2004). *Floodplain risk analysis using flood probability and annual exceedance probability maps*. Ph.d. diss., Brigham Young University.

US Army Corps of Engineers Hydrologic Engineering Center (1998). *HEC-1: Flood Hydrograph Package User's Manual*. US Army Corps of Engineers, Hydrologic Engineering Center.

US Army Corps of Engineers Hydrologic Engineering Center (2001). *HEC-RAS River Analysis System: User's Manual*. US Army Corps of Engineers, Hydrologic Engineering Center.

US Army Corps of Engineers Hydrologic Engineering Center (2003). *HEC-GeoHMS Geospatial Hydrologic Modeling Extension: User's Manual*. US Army Corps of Engineers, Hydrologic Engineering Center.

Williams, S. & Kindel, C. (1994). *The Component Object Model: A Technical Overview*. Microsoft Corporation.

Zeiler, M. (1999). *Modeling Our World: The ESRI Guide to Geodatabase Design*. ESRI Press.

Zeiler, M. (2001). *Exploring ArcObjects*. ESRI Press.



www.manaraa.com

# APPENDIX



www.manaraa.com

www.manaraa.com

# A C++ Source Code

This appendix includes source code for creating geodatabases consisting of feature datasets, feature classes, and its accompanying terrain data using ArcObjects. Each part will describe what the source code does, and the prerequisite data required to set it up. The source code assumes that the prerequisite data is initialized by the programmer, and can be modified and adapted for use in any program to write out a geodatabase. See Zeiler (2001) for a description and use of the ESRI variable types.

## A.1 Using ArcObjects in Visual C++

In order to use ArcObjects inside the Visual C++ programming environment, an installation of ArcView 8 or higher is required. The object libraries are installed along with ArcView. In order to access ArcObjects on a particular computer, a valid license to the ESRI products must be checked out. A program using ArcObjects code can be compiled without a license to ArcView as long as the object library is copied onto the computer compiling the program. Without a license, however, none of the ArcObjects code can be accessed. Table A.1 shows what is needed to use the ArcObjects object library, and Figure A.1 shows the C++ code used to access the object library.

Table A.1: Requirements for the ESRI object libraries.

| **Item** | |
|---|---|
| ArcObjects object libraries | esriCore.olb for ArcView 8 |
| Valid license | (to access ArcObjects code) |



www.manaraa.com

## A.2 Creating a Spatial Reference

A common spatial reference is required for each feature dataset in the geodatabase. Spatial references place a dataset in its proper coordinate system. The following code shows how to create spatial references for geographic and projected coordinate systems. Tables A.2 and A.3 show what is necessary to create geographic and projected coordinate system spatial references, while Figures A.2 and A.3 show the C++ code for creating the spatial references.

### A.2.1 Creating a geographic coordinate system

A geographic coordinate system is one way of defining a spatial reference.

Table A.2: Requirements for a geographic coordinate system.

| **Item** | **Variable type** |
|---|---|
| Type of coordinate system<br>(Such as NAD83, Clark 1880, WGS 1984) | esriSRGeoCSType |

### A.2.2 Creating a projected coordinate system

A projected coordinate system contains a geographic coordinate system, a project, and other parameters. A spatial reference can also be defined from a projected coordinate system.

## A.3 Creating a Personal Geodatabase Workspace Factory

A Workspace Factory is necessary for creating a geodatabase. A Workspace Factory can be initialized to create enterprise geodatabases for Oracle, Sybase, Ingress or other systems, or a personal geodatabase system for Microsoft Access databases. Figure A.4 shows the



www.manaraa.com

```
#import <esriCore.olb> raw_interfaces_only,\
                       named_guids,\
                       no_namespace,\
                       exclude("OLE_HANDLE", "OLE_COLOR")
```

Figure A.1: Accessing the ESRI object libraries for ArcView 8.

```
esriSRGeoCSType                 gtype;
ISpatialReferenceFactory2Ptr    ipFact;
IGeographicCoordinateSystemPtr  ipGCS;
ISpatialReferencePtr            ipSpatialRef;

ipFact.CreateInstance(CLSID_SpatialReferenceEnvironment);
ipFact->CreateGeographicCoordinateSystem(gtype, &ipGCS);
ipSpatialRef = ipGCS;
ipSpatialRef->SetFalseOriginAndUnits(-180.0, -90.0, 1000000.0);
```

Figure A.2: Creating a geographic coordinate system spatial reference.

Table A.3: Requirements for a projected coordinate system.

| **Item** | **Variable type** |
|---|---|
| Type of coordinate system<br><br>(Such as NAD83, Clark 1880, WGS 1984) | esriSRGeoCSType |

```
esriSRProjCSType                ptype;
ISpatialReferenceFactory2Ptr    ipFact;
ISpatialReferencePtr            ipSpatialRef;

ipFact.CreateInstance(CLSID_SpatialReferenceEnvironment);
ipFact->CreateProjectedCoordinateSystem(ptype, &ipPCS);
ipSpatialRef = ipPCS;
ipSpatialRef->SetFalseOriginAndUnits(0.0, 0.0, 1.0);
```

Figure A.3: Creating a projected coordinate system spatial reference.



www.manaraa.com

source code to create a Microsoft Access workspace factory.

```
IWorkspaceFactoryPtr ipWorkspaceFactory;

ipWorkspaceFactory.CreateInstance(CLSID_AccessWorkspaceFactory);
```

Figure A.4: Creating a Microsoft Access workspace factory.

## A.4 Creating a Workspace

Creating a workspace starts a new geodatabase file on disk. ArcObjects allows the user to specify the path and filename of the new geodatabase. Table A.4 shows the required data to create a workspace, and Figure A.5 shows the C++ to create it.

Table A.4: Requirements for a workspace.

| **Item** | **Variable type** |
|---|---|
| Workspace factory | IWorkspaceFactory |
| Path of the workspace | CComBSTR |
| Filename of the workspace | CComBSTR |

## A.5 Editing a Workspace

In order to add data to the geodatabase, ArcObjects must be first used to enable editing. Only a valid workspace is required, as shown in Table A.5. The C++ code to start editing the workspace is shown in Figure A.6.

## A.6 Creating a Feature UID

A UID (unique identifier object) must be first created in order to later create feature classes within feature datasets. Code for creating a UID is shown in Figure A.7.



www.manaraa.com

```
IWorkspacePtr          ipWorkspace;
CComBSTR               szPath, szFile;
IWorkspaceNamePtr      ipWorkspaceName;
INamePtr               ipName;
IUnknownPtr            ipUnk;
IPropertySetPtr        ipProperty;
IWorkspaceFactoryPtr   ipFact;

// Create the workspace property set
ipProperty.CreateInstance(CLSID_PropertySet);

// Using the workspace factory, create a new workspace,
// with the properties stored in the workspace name interface
ipFact->Create(szPath, szFile, ipProperty, 0, &ipWorkspaceName);

// Open the workspace
ipName = ipWorkspaceName;
hr = ipName->Open(&ipUnk);
ipWorkspace = ipUnk;
```

Figure A.5: Creating and opening a workspace.

Table A.5: Requirements for editing a workspace.

| **Item** | **Variable type** |
|---|---|
| Workspace | IWorkspace |

```
IWorkspacePtr     ipWorkspace;
IWorkspaceEditPtr ipWorkspaceEdit;

ipWorkspaceEdit = ipWorkspace;

// Start editing, disabling the undo/redo logging
ipWorkspaceEdit->StartEditing(VARIANT_FALSE);
```

Figure A.6: Creating and opening a workspace.



www.manaraa.com

## A.7 Creating ArcHydro Geodatabase Domains

Domains are used to define ranges and coded values, for validation processes in the geodatabase and for easier display and entry of data into geodatabase fields. Domains for the ArcHydro data model are listed in Maidment (2002). Table A.6 shows what is required to create a domain for a workspace, while Figure A.8 shows how to create the ArcHydro FlowDomain using C++.

Table A.6: Requirements for creating a domain.

| **Item** | **Variable type** |
|---|---|
| Workspace | IWorkspace |
| Domain values | int, float, string, etc. |

## A.8 Creating Feature Datasets

Feature datasets are used to store feature classes within a geodatabase. Each of the feature classes stored in a feature dataset should have a common spatial reference, which is stored at the feature dataset level. If following the ArcHydro data model, feature datasets should be created for the Channel, Drainage, Hydrography and Network feature classes. Table A.7 shows the data required to build a feature dataset. Figure A.9 shows how to create a feature dataset, once the spatial reference has been set up.

## A.9 Creating Fields for Feature Classes

Once a feature dataset has been created, the feature classes that are to be stored in it can be created. In order to create the feature classes, however, the fields comprising each feature class need to be created. See Maidment (2002) for the field definitions for each feature class in the ArcHydro data model.



www.manaraa.com

```
IUIDPtr  ipUid;

// Use the CLSID of esriCore.Feature
CComVariant vVariant("{52353152-891A-11D0-BEC6-00805F7C4268}");

// Create the UID from the string holding info about ESRI features
ipUid.CreateInstance(CLSID_UID);
ipUid->put_Value(vVariant);
```

Figure A.7: Creating a feature UID.

Table A.7: Requirements for creating a feature dataset.

| **Item** | **Variable type** |
|---|---|
| Workspace | IWorkspace |
| Spatial reference | ISpatialReference |
| The name of the feature dataset | CComBSTR |



www.manaraa.com

```
IWorkspacePtr          ipWorkspace;
IDomainPtr             ipDomain;
ICodedValueDomainPtr   ipCVDomain;
IWorkspaceDomainsPtr   ipWSDomains;
CComVariant            vValue;
long                   lID;

// Create the domain
ipCVDomain.CreateInstance(CLSID_CodedValueDomain);

// Add the coded values to the domain
vValue = 1;  vValue.ChangeType(VT_I4);
ipCVDomain->AddCode(vValue, CComBSTR("Uninitialized"));
vValue = 1;  vValue.ChangeType(VT_I4);
ipCVDomain->AddCode(vValue, CComBSTR("WithDigitized"));
vValue = 2;  vValue.ChangeType(VT_I4);
ipCVDomain->AddCode(vValue, CComBSTR("AgainstDigitized"));
vValue = 3;  vValue.ChangeType(VT_I4);
ipCVDomain->AddCode(vValue, CComBSTR("Indeterminate"));

// Set up the other domain parameters
ipDomain = ipCVDomain;
ipDomain->put_Name(CComBSTR("HydroFlowDirections"));
ipDomain->put_FieldType(esriFieldTypeInteger);
ipDomain->put_MergePolicy(esriMPTDefaultValue);
ipDomain->put_SplitPolicy(esriSPTDefaultValue);

// Adding the domain to the workspace
ipWSDomains = ipWorkspace;
ipWSDomains->AddDomain(ipDomain, &lID);
```

Figure A.8: Creating the HydroFlow domain and adding it to the workspace.



www.manaraa.com

### A.9.1 Creating the basic fields for a feature class

Each feature class requires two fields at a minimum. These are the object ID field, and the shape field, which describes the type of feature to be stored. Table A.8 shows the data needed to create the required fields of a simple feature class, and Figure A.10 shows how to set up the fields for a point, polyline or polygon feature class.

Table A.8: Requirements for creating basic feature class fields.

| **Item** | **Variable type** |
|---|---|
| Shape type<br>(Such as point, polyline, polygon) | esriGeometryType |
| Whether the shapes will hold Z values | VARIANT_BOOL |
| Whether the shapes will hold M values | VARIANT_BOOL |

### A.9.2 Adding additional fields for a feature class

Once the required fields are created, additional attribute fields can be added, such as those defined in the ArcHydro data model. Table A.9 shows the data needed to define additional attribute fields for a feature class. Figure A.11 demonstrates how to add an additional field using C++.

## A.10 Creating Feature Classes

Once the fields for a feature class have been made, the feature class can be created. Normally, feature classes are created inside of feature datasets, but that can also be created in the root of the geodatabase. Table A.10 show the data required to create a feature class, and Figure A.12 shows how to build a new feature class in a feature dataset.



www.manaraa.com

```
IWorkspacePtr          ipWorkspace;
IFeatureWorkspacePtr   ipFeatureWorkspace;
ISpatialReferencePtr   ipSpatialRef;
IFeatureDatasetPtr     ipFeatureDataset;
CComBSTR               szName;

ipFeatureWorkspace = ipWorkspace;

ipFeatureWorkspace->CreateFeatureDataset(szName,
                                         ipSpatialRef,
                                         &ipFeatureDataset);
```

Figure A.9: Creating a feature dataset.

Table A.9: Requirements for creating additional feature class fields.

| **Item** | **Variable type** |
|---|---|
| The field name | CComBSTR |
| Field type<br>(Such as double, string, etc.) | esriFieldType |
| Field default value | long, double, string, etc. |
| Whether the field is nullable | VARIANT_BOOL |
| The domain of the field | IDomain |
| The field's precision<br>(for numerical fields) | long |
| The field's scale<br>(for floating point fields) | long |
| The length of the field<br>(for string fields) | long |
| The required feature class fields | IFieldsEdit |



www.manaraa.com

```
esriGeometryType     geom;
VARIANT_BOOL         bZ, bM;
IFieldsEditPtr       ipFieldsEdit;
IFieldEditPtr        ipFieldEdit;
IGeometryDefEditPtr  ipGeomDef;
long                 numpoints;

ipFieldsEdit.CreateInstance(CLSID_Fields);
// Create the Object ID Field
ipFieldEdit.CreateInstance(CLSID_Field);
ipFieldEdit->put_Name(CComBSTR("OID"));
ipFieldEdit->put_Type(esriFieldTypeOID);
ipFieldEdit->put_AliasName(CComBSTR("Object ID"));
ipFieldEdit->put_IsNullable(VARIANT_FALSE);
ipFieldsEdit->AddField(ipFieldEdit);

// Create the geometry definition
ipGeomDef.CreateInstance(CLSID_GeometryDef);
switch (geom) {
  case esriGeometryPoint:     numpoints = 1;  break;
  case esriGeometryPolyline:  numpoints = 5;  break;
  case esriGeometryPolygon:   numpoints = 20; break;
  default:                    numpoints = 1;  break;
}
ipGeomDef->put_AvgNumPoints(numpoints);
ipGeomDef->put_GeometryType(geom);
ipGeomDef->put_GridCount(1);
ipGeomDef->put_GridSize(0, 100);
ipGeomDef->put_HasM(bM);
ipGeomDef->put_HasZ(bZ);

// Create the Shape (geometry) field
ipFieldEdit.CreateInstance(CLSID_Field);
ipFieldEdit->put_Name(L"Shape");
ipFieldEdit->put_Type(esriFieldTypeGeometry);
ipFieldEdit->put_IsNullable(VARIANT_TRUE);
ipFieldEdit->put_Editable(VARIANT_TRUE);
ipFieldEdit->put_AliasName(L"Shape");
ipFieldEdit->putref_GeometryDef(ipGeomDef);
ipFieldsEdit->AddField(ipFieldEdit);
```

Figure A.10: Creating basic feature class fields.



www.manaraa.com

```
CComBSTR       szName;
esriFieldType  eType;
VARIANT_BOOL   bNullable;
long           lPrecision;
long           lLength;
long           lDefault;
long           lScale;
IDomainPtr     ipDomain;
IFieldsEditPtr ipFieldsEdit;
IFieldEditPtr  ipFieldEdit;
CComVariant    vDefault(lDefault);

ipFieldEdit.CreateInstance(CLSID_Field);
FieldEdit->put_Name(szName);
ipFieldEdit->put_Type(eType);
ipFieldEdit->put_AliasName(CComBSTR(szName));
ipFieldEdit->put_IsNullable(bNullable);
ipFieldEdit->put_Precision(lPrecision);
ipFieldEdit->put_DefaultValue(vDefault);
ipFieldEdit->put_Length(lLength);
if (ipDomain != 0) {
  hr = ipFieldEdit->putref_Domain(ipDomain);
ipFieldsEdit->AddField(ipFieldEdit);
```

Figure A.11: Creating a feature class attribute field.

Table A.10: Requirements for creating a simple feature class.

| **Item** | **Variable type** |
|---|---|
| The feature class name | CComBSTR |
| The feature dataset to store it in | IFeatureDataset |
| The fields of the feature class | IFieldsEdit |
| The feature UID | IUID |
| The name of the shape field (usually shape) | CComBSTR |



www.manaraa.com

## A.11 Creating Features

Once a feature class has been created, individual features can be added to it. This section demonstrates how to add the shape information to point, polyline and polygon features.

### A.11.1 Point features

Point features are easy to define, as they only require an XY location, as shown in Table A.11. Additional attributes, such as the Z value or M measure can also be added. Figure A.13 shows how to create and define a new point.

Table A.11: Requirements for creating a point feature.

| **Item** | **Variable type** |
|---|---|
| XY location | double |
| Z value (optional) | double |
| M value (optional) | double |

### A.11.2 Polyline features

Polyline features are also relatively easy to define, as they only require an XY location on each section defining the polyline, as shown in Table A.12. Additional attributes, such as the Z value or M measure can also be added. Figure A.14 shows how to create and define a new polyline.



www.manaraa.com

```
CComBSTR            szName;
IFeatureDatasetPtr  ipFeatureDataset;
IFieldsEditPtr      ipFieldsEdit;
IUIDPtr             ipUID;
IFeatureClassPtr    ipFeatureClass;

ipFeatureDataset->CreateFeatureClass(szName, ipFieldsEdit,
                                     ipUID, 0, esriFTSimple,
                                     CComBSTR("Shape"),
                                     CComBSTR(""),
                                     &ipFeatureClass);
```

Figure A.12: Creating a simple feature class.

```
int        bHasZ, bHasM;
double     x, y, z, m;
IPointPtr  ipPoint;
IZAwarePtr ipZAware;
IMAwarePtr ipMAware;

ipPoint.CreateInstance(CLSID_Point);

// Set up the Z and/or M attributes for the point
if (bHasZ) {
  ipZAware = ipPoint;
  hr = ipZAware->put_ZAware(VARIANT_TRUE);
}
if (bHasM) {
  ipMAware = ipPoint;
  hr = ipMAware->put_MAware(VARIANT_TRUE);
}

hr = ipPoint->put_X(x);
hr = ipPoint->put_Y(y);
if (bHasZ)
  hr = ipPoint->put_Z(z);
if (bHasM)
  hr = ipPoint->put_M(m);
```

Figure A.13: Creating a point feature.



www.manaraa.com

```
IPointPtr            ipPoint;
IPointCollectionPtr  ipPointCollection;
IPolylinePtr         ipPolyline;
IZAwarePtr           ipZAware;
IMAwarePtr           ipMAware;
double               *x, *y, *z, *m;
int                  i, numpoints, bHasZ, bHasM;

ipPoint.CreateInstance(CLSID_Point);
ipPointCollection.CreateInstance(CLSID_Polyline);

// Set up the Z and/or M attributes for the point
if (bHasZ) {
  ipZAware = ipPoint;
  hr = ipZAware->put_ZAware(VARIANT_TRUE);
}
if (bHasM) {
  ipMAware = ipPoint;
  hr = ipMAware->put_MAware(VARIANT_TRUE);
}

// Add the points
for (i=0; i<numpoints; i++) {
  ipPoint->put_X(x[i]);
  ipPoint->put_Y(y[i]);
  if (bHasZ)
    ipPoint->put_Z(z[i]);
  if (bHasM)
    ipPoint->put_M(m[i]);
  hr = ipPointCollection->AddPoint(ipPoint);
}
ipPolyline = ipPointCollection;
if (bHasZ) {
  ipZAware = ipPolyline;
  hr = ipZAware->put_ZAware(VARIANT_TRUE);
}
if (bHasM) {
  ipMAware = ipPolyline;
  hr = ipMAware->put_MAware(VARIANT_TRUE);
}
```

Figure A.14: Creating a polyline feature.



www.manaraa.com

### A.11.3 Polygon features

Polyline features are more complex than points and polylines, but they too require the XY locations along their boundary, as shown in Table A.13. Figure A.15 shows how to create and define a new polyline consisting of one outer ring only.

```
IPointPtr                ipPoint;
IPointCollectionPtr      ipPointCollection;
IGeometryCollectionPtr   ipGeometryCollection;
ITopologicalOperatorPtr  ipTopologicalOperator;
IRingPtr                 ipRing;
IPolygonPtr              ipPolygon;
IGeometryPtr             ipGeometry;
int                      i, numpoints;
double                   *x, *y;

ipPoint.CreateInstance(CLSID_Point);
ipPointCollection.CreateInstance(CLSID_Ring);
ipGeometryCollection.CreateInstance(CLSID_Polygon);

for (i=0; i<numpoints; i++) {
  ipPoint->put_X(x[i]);
  ipPoint->put_Y(y[i]);
  ipPointCollection->AddPoint(ipPoint);
}

// Set up the geometry
ipRing = ipPointCollection;
ipGeometry = ipRing;
ipGeometryCollection->AddGeometry(ipGeometry);

// Put the geometry into a polygon and simplify
ipPolygon = ipGeometryCollection;
ipTopologicalOperator = ipPolygon;
ipTopologicalOperator->Simplify();
```

Figure A.15: Creating a polygon feature.



www.manaraa.com

Table A.12: Requirements for creating a polyline feature.

| **Item** | **Variable type** |
|---|---|
| XY location of each point on line | double |
| Z value of each point on line (optional) | double |
| M value (optional) | double |

Table A.13: Requirements for creating a polygon feature.

| **Item** | **Variable type** |
|---|---|
| XY location of each point on polygon | double |



www.manaraa.com

### A.11.4 Storing a feature in the feature class

Once a new point, polyline or polygon feature is created and defined, it needs to be stored in it's feature class. Table A.14 shows the requirements to store a new feature in a feature class. Figure A.14 shows the C++ code to store a polyline in a polyline feature class, though the process is similar for point and polygon feature classes.

Table A.14: Requirements for storing a feature in a feature class.

| **Item** | **Variable type** |
|---|---|
| Feature class | IFeatureClass |
| Point, polyline or polygon | IPoint, IPolyline, IPolygon |

```
IPolylinePtr     ipPolyline;
IFeatureClassPtr ipFeatureClass;
IFeaturePtr      ipFeature;

ipFeatureClass->CreateFeature(&ipFeature);
ipFeature->putref_Shape(ipLine);
ipFeature->Store();
```

Figure A.16: Storing a polyline feature in a feature class.

## A.12 Filling Attribute Fields

After the geometry of each feature in a feature class has been defined, the attributes can be stored in the fields, such as those defined for the ArcHydro data model. Table A.15 shows the data required to fill in an attribute field for a feature. Figure A.17 shows how to fill in the attributes of a polygon feature, given an array of values for each of the attribute fields.



www.manaraa.com

Table A.15: Requirements for filling attribute fields for a feature.

| **Item** | **Variable type** |
|---|---|
| Feature<br><br>(with previously defined fields) | IFeature |
| Field attribute data | double, int, string, etc. |

```
IFeaturePtr  ipFeature;
IFieldsPtr   ipFields;
IFieldPtr    ipField;
long         lNumFields, i;
CComBSTR     bFieldName;
CComVariant  *vValues;  // Attributes for each field

ipFeature->get_Fields(&ipFields);
// Get the field count, and skip the OID and Shape fields
ipFields->get_FieldCount(&lNumFields);
for (i=2; i<lNumFields; i++) {
  ipFields->get_Field(i, &ipField);
  ipFeature->put_Value(i, vValues[i]);
}
```

Figure A.17: Filling in attributes for a polygon feature.



www.manaraa.com

## A.13 Creating Stand Alone Tables

Stand alone tables, such as the HydroID table in the ArcHydro data model, are handled similarly to the attribute tables for feature classes. However, the base object is a table instead of a feature. The fields are created in the same manner as for features, except that there is no field to hold the geometry of the shape. Table A.16 shows the data required to fill in the fields of a stand alone table, which is similar to the data needed for a feature's attribute fields. Figure A.18 shows how to fill in the rows of a stand alone table, given an array of values for each of the attribute fields.

Table A.16: Requirements for filling fields of a stand alone table.

| **Item** | **Variable type** |
|---|---|
| Table<br>(with previously defined fields) | ITable |
| Field attribute data | double, int, string, etc. |

## A.14 Creating a Raster From Elevation Data

A raster dataset, such as an ESRI GRID, can be used to store information from a DEM or other gridded elevation source. An ESRI GRID can be stored in an enterprise geodatabase or separately from a personal geodatabase, for inclusion with a FEMA project. Table A.17 shows the data required to set up a new raster. Figure A.19 shows C++ code for first setting up a raster dataset, and Figure A.20 shows how to fill in the dataset with an array of elevation values.



www.manaraa.com

```
ITablePtr    ipTable;
IRowPtr      ipRow;
IFieldsPtr   ipFields;
IFieldPtr    ipField;
long         lNumFields, i;
CComBSTR     bFieldName;
CComVariant  *vValues;  // Attributes for each field

ipTable->get_Fields(&ipFields);
// Get the field count, and skip the OID
ipFields->get_FieldCount(&lNumFields);
for (i=1; i<lNumFields; i++) {
  ipTable->CreateRow(&ipRow);
  ipFields->get_Field(i, &ipField);
  ipRow->put_Value(i, vValues[i]);
}
```

Figure A.18: Filling in attributes for a polygon feature.

Table A.17: Requirements for creating a raster elevation grid.

| **Item** | **Variable type** |
|---|---|
| Gridded elevation data | array of float or double values |
| Number of rows and columns in grid | integer |
| Cell size | double |
| Grid XY origin | double |
| Spatial reference | ISpatialReference |
| Path to save the grid | CComBSTR |
| Workspace name | CComBSTR |
| Grid name | CComBSTR |
| Grid format<br>(GRID, TIFF, IMAGINE Image) | CComBSTR |



www.manaraa.com

```
double                    cellsize;
int                       numrows, numcols;
IPointPtr                 ipOrigin;
double                    x, y;
ISpatialReferencePtr      ipSpatialRef;
IWorkspacePtr             ipWorkspace;
IWorkspaceFactoryPtr      ipFactory;
CComBSTR                  szName, szPath, szWorkspaceName;
CComBSTR                  szGridType;
IPropertyPtr              ipPropery;
INamePtr                  ipName;
IWorkspaceNamePtr         ipWorkspaceName;
IUnknownPtr               ipUnknown;
IRasterWorkspace2Ptr      ipRasterWorkspace;
IRasterDatasetPtr         ipRasterDataset;

// Create raster workspace
ipFactory.CreateInstance(CLSID_RasterWorkspaceFactory);
ipProperty.CreateInstance(CLSID_PropertySet);
ipFactory->Create(szPath, szWorkspaceName, ipProperty, 0,
                  &ipWorkspaceName);
ipName = ipWorkspaceName;
ipName->Open(&ipUnknown);
ipWorkspace = ipUnknown;
ipRasterWorkspace = ipWorkspace;

// Create raster dataset
ipOrigin.CreateInstance(CLSID_Point);
ipOrigin->PutCoords(x, y);
ipRasterWorkspace->CreateRasterDataset(szName, szGridType,
                                       ipOrigin, numcols,
                                       numrows, cellsize,
                                       cellsize, 1, PT_FLOAT,
                                       ipSpatialRef,
                                       &ipRasterDataset);
```

Figure A.19: Creating a raster dataset.



www.manaraa.com

```
float                       **elevation;
double                      cellsize;
long                        numrows, numcols, i, j k, lWidth, lHeight;
IRasterDatasetPtr           ipRasterDataset;
IRasterPtr                  ipRaster;
IRasterBandCollectionPtr    ipBandCollection;
IRasterBandPtr              ipRasterBand;
IRasterPropsPtr             ipRasterProps;
IRawPixelsPtr               ipRawPixels;
IPntPtr                     ipSize, ipPnt;
IPixelBlockPtr              ipPixelBlock;
CComVariant                 vSafeArray;
SAFEARRAY                   *saArray;
float HUGEP                 *fArray = NULL;

// Create raster and raster band
ipRasterDataset->CreateDefaultRaster(&ipRaster);
ipRasterProps = ipRaster;
ipBandCollection = ipRaster;
ipBandCollection->Item(0, &ipRasterBand);
ipRawPixels = ipRasterBand;
ipSize.CreateInstance(CLSID_DblPnt);
ipRasterProps->get_Width(&lWidth);
ipRasterProps->get_Height(&lHeight);
ipSize->SetCoords((double)lWidth, (double)lHeight);
ipRawPixels->CreatePixelBlock(ipSize, &ipPixelBlock);

// Get the SafeArray and set elevation values
ipPixelBlock->get_SafeArray(0, &vSafeArray);
saArray = vSafeArray.parray;
::SafeArrayAccessData(saArray, (void HUGEP* FAR*)&fArray);
for (i = 0, k = 0; i < lHeight; i++) {
  for (j = 0; j < lWidth; j++, k++)
    fArray[k] = elevation[i][j];
}
::SafeArrayUnaccessData(saArray);
ipPnt.CreateInstance(CLSID_DblPnt);
ipPnt->SetCoords(0.0, 0.0);
ipRawPixels->Write(ipPnt, ipPixelBlock);
```

Figure A.20: Creating a raster band and setting elevation values.



www.manaraa.com

www.manaraa.com