DOTAGWA: A WATERSHED ASSESSMENT TOOL IN NATURAL RESOURCES
INFORMATION SYSTEMS

by

Averill Cate, Jr.

A Document Submitted to the Faculty of the
SCHOOL OF NATURAL RESOURCES
In Partial Fulfillment of the Requirements
For the Degree of
DOCTOR OF PHILOSOPHY
WITH A MAJOR IN RENEWABLE NATURAL RESOURCES STUDIES

In the Graduate College

THE UNIVERSITY OF ARIZONA

2008
As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Averill Cate, Jr. entitled

DOTAGWA: A WATERSHED ASSESSMENT TOOL IN NATURAL RESOURCES INFORMATION SYSTEMS

and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy

_______________________________________________________________________ Date: 05/07/2008

D. Phillip Guertin

_______________________________________________________________________ Date: 05/07/2008

David C. Goodrich

_______________________________________________________________________ Date: 05/07/2008

Stuart E. Marsh

_______________________________________________________________________ Date: 05/07/2008

Randy H. Gimblett

Final approval and acceptance of this dissertation is contingent upon the candidate’s submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

_______________________________________________________________________ Date: 05/07/2008

Dissertation Director: D. Phillip Guertin
STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Averill Cate, Jr.
ACKNOWLEDGEMENTS

The following body of work represents work completed in pursuit of my educational training in the application of Geographic Information Science for Natural Resources. Many people have helped me along the way. I’d like to acknowledge my dissertation committee. First, I’d like to thank Phil Guertin. Many years ago he helped me start my pursuit of this degree and since then I have learned many things both academically and personally. Second, I’d like to thank Dave Goodrich for inviting me to work on the DotAGWA project. I have loved every minute of it and have learned many things from Dave. Lastly, I’d like to thank Stuart Marsh and Randy Gimblett for being great instructors and for being incredibly friendly people. Being in academics puts tremendous strain on a person’s time. Quite often instructors and even advisors can be hard to reach due to other commitments. This was not the case for me. All of my committee advisors were very accessible both academically and maybe more importantly as friends. Finally, I’d like to thank the Southwest Watershed Research Center, the United States Department of Agriculture’s Agricultural Research Service, and the United States Environmental Protection Agency for the financial supporting that made my research project possible.
# TABLE OF CONTENTS

LIST OF ILLUSTRATIONS .............................................................................................. 7
ABSTRACT ....................................................................................................................... 8
INTRODUCTORY OVERVIEW ........................................................................................ 10
  Context of the Research ............................................................................................. 10
  History of AGWA ....................................................................................................... 14
  History of DotAGWA ............................................................................................... 15
PRESENT STUDY ........................................................................................................... 18
  Overview of the Three Papers ................................................................................... 18
REFERENCES ................................................................................................................. 21

APPENDIX A MANUSCRIPT 1: DESIGN OF THE INTERNET BASED AUTOMATED GEOSPATIAL WATERSHED ASSESSMENT (AGWA) TOOL, DOTAGWA ...................................................................................................................... 22
  ABSTRACT .................................................................................................................. 23
  INTRODUCTION ........................................................................................................ 25
  METHODS ................................................................................................................... 29
    System Design .......................................................................................................... 29
    User and System Requirements ................................................................................ 30
    Primary Use Cases .................................................................................................... 30
    Secondary Use Cases ............................................................................................... 31
    Activity Diagrams ..................................................................................................... 36
    System Interaction Diagrams .................................................................................... 50
  DISCUSSION ............................................................................................................... 57
  CONCLUSIONS .......................................................................................................... 62
  LITERATURE CITED ................................................................................................. 64

APPENDIX B MANUSCRIPT 2: DOTAGWA: A CASE STUDY IN INTERNET BASED ARCHITECTURES FOR CONNECTING SURFACE WATER MODELS TO SPATIALLY ENABLED WEB APPLICATIONS ............................................................................... 66
  ABSTRACT .................................................................................................................. 67
  INTRODUCTION ........................................................................................................ 69
  APPLICATION DESIGN ............................................................................................. 71
    Application Users ...................................................................................................... 71
    System Requirements ............................................................................................... 72
    Software .................................................................................................................... 72
    Programming Environment ....................................................................................... 72
    Database Server ........................................................................................................ 73
    Web Server ............................................................................................................... 73
    Spatial Data Server .................................................................................................. 73
  Data ........................................................................................................................... 74
  Hardware ................................................................................................................... 74
  APPLICATION DEVELOPMENT FINDINGS ................................................................ 76
    Application Development ......................................................................................... 76
    Interface Development ............................................................................................ 78
LIST OF ILLUSTRATIONS

Figure 1 AGWA System Navigation ................................................................. 27
Figure 2 AGWA Primary Use Cases ............................................................... 31
Figure 3 Plan Development - Secondary Use Case ........................................... 32
Figure 4 Plan Management Use Cases ............................................................ 33
Figure 5 Delineator Use Case ........................................................................ 34
Figure 6 KINEROS Elements ....................................................................... 35
Figure 7 Discretization Use Case ................................................................. 36
Figure 8 Login Activity ................................................................................ 38
Figure 9 Delineation Activity ....................................................................... 39
Figure 10 Discretization Activity ................................................................. 41
Figure 11 Land Cover Parameterization ....................................................... 43
Figure 12 Parameterization Activity ............................................................ 45
Figure 13 Land Cover Parameterization ....................................................... 46
Figure 14 Run Model Activity .................................................................... 48
Figure 15 View Output Activity .................................................................. 49
Figure 16 Delineation Sequence .................................................................. 51
Figure 17 Discretization Sequence ............................................................. 52
Figure 18 Soils Parameterization Sequence ................................................ 53
Figure 19 Land Cover Parameterization Sequence ...................................... 54
Figure 20 Run Model Sequence .................................................................. 55
Figure 21 View Output Sequence ................................................................ 56
Figure 22 Basic System Architecture ............................................................ 77
Figure 23 Basic Work Flow ......................................................................... 79
Figure 24 DotAGWA Main Interface ............................................................. 80
Figure 25 User Selected Point for Watershed Delineation ................................ 81
Figure 26 Initializing Model Run ................................................................. 82
Figure 27 Model Output ............................................................................. 83
Figure 28 Plan Management ....................................................................... 84
Figure 29 Database Models and Relationships ............................................ 88
Figure 30 Simple Use Case Diagram ........................................................... 102
Figure 31 Model Input Reader Strategy Design Pattern ............................... 107
Figure 32 Output Processing Strategy Design Pattern ................................. 109
Figure 33 Decorator Pattern Applied to Management Plans ........................ 110
ABSTRACT

The practice of linking Geographic Information Systems (GIS) software and natural resource models has greatly increased in the recent past. Cheaper and more powerful computing resources have allowed us to build systems that minimize the effort and labor involved in parameterizing simulation models. However, by using computerized means to minimize the effort needed to facilitate model parameterization we have increased the complexity in these links between the two components. We have also increased the amount of knowledge required to build the link and have increased the need to understand the consequences of building the links between two systems. The practice of linking these two components creates new issues that affect both the GIS analyst and the researcher (Hartkamp et al., 1999). The goal of this research project has been to develop an application linking GIS-based geo-processing tools developed in the Automated Geospatial Watershed Assessment (AGWA) tool (Miller et al., 2007) to an internet-based map interface. The application allows a user to develop a management scenario by delineating a watershed based on one or more outlet points. The application uses the delineation and other input data sets to develop input parameter files for a hydrologic model, which then runs and produces output for the user. The development of the application produced many interesting issues, but the one identified as most important in terms of this dissertation research was an issue related to using current software development tools such as the Universal Modeling Language (UML) and software design patterns as a way to communicate about system requirements and system functions.
between programmers and project stakeholders. This research will examine how these software development tools were used to develop DotAGWA, the consequences of using the tools and an analysis of why these tools may be an important component in developing natural resource projects that rely heavily on GIS tools.
INTRODUCTORY OVERVIEW

Context of the Research

Linking GIS applications and hydrologic models has gained importance with increasing demands on natural resources and increasing desires by communities to preserve and even enhance natural environments. Hydrologic models and desktop-GIS applications have existed for a very long time. Traditionally, hydrologic models were parameterized manually by modelers having to collect field data, clean the data, analyze the data, and transform the data into an electronic format. Once the data were in electronic format they still needed further formatting work to make them suitable for hydrological modeling. All of this labor was very time intensive, vulnerable to error, and difficult to keep organized.

The advent of GIS software has greatly facilitated the formatting and organization required to manage the input data used in hydrologic modeling (Hartkamp et al., 1999). GIS software has also created powerful display tools allowing researchers to visualize their data much more readily. GIS software has provided extremely useful tools that help researchers automate data formatting processes and organization. Given these advancements created by employing GIS software researchers naturally began to look towards ways to build links between GIS software and hydrologic models. Generally, the goals of building applications that link hydrologic models to GIS applications has
centered on determining how to help automate the model parameterization, automate model execution, and automate manipulation and display of model output (Miller et al., 2007).

The Automated Geospatial Watershed Assessment tool (AGWA) is a GIS application that serves as a link between two hydrologic models, SWAT and KINEROS, and the supporting soils, land cover, and climate data required to run these hydrologic models. A primary goal of the AGWA project was to provide a desktop tool for conducting all phases of a watershed assessment for SWAT and KINEROS (Miller et al., 2007).

Current trends show communities are interested in understanding the relationships between land use and their watersheds and associated resources. However, some of these people may not have access to the GIS software, data, and models required to construct watershed analyses. Furthermore, people needing access to tools like AGWA may not want to spend the time required to become a hydrologist and/or GIS expert. Given these conditions, DotAGWA was developed as an internet-based version of the AGWA application. DotAGWA allows users with connections to the internet to connect to DotAGWA via a web client (e.g., Internet Explorer) and through an interface that presents maps of an area of interest, allows the user to delineate a watershed, describe a management plan, discretize a model for the delineated watershed and management plan, parameterize that model and run it with the hydrologic model’s output, presenting text and graphical output to the user.
Research presented in this dissertation was completed and based on the development of the DotAGWA application. The author was the lead developer of DotAGWA and this dissertation presents the body of knowledge discovered during the development of this application. DotAGWA was developed to create a watershed assessment tool that, by using the Internet as the delivery mechanism, might make GIS applications and hydrologic models available to a wider range of users than the equivalent standalone desktop application could. This dissertation will discuss the benefits and weaknesses of different key aspects discovered during the planning and development stages of the application and discuss important findings revealed during these same project development stages.

In particular this dissertation includes research related to the use and impact in DotAGWA application development of the Unified Modeling Language (UML) and software design patterns. UML is a standard for object oriented modeling notation (Jia, 2002) used in software development and other industries. UML was originally developed by three methodologists named Grady Booch, Ivar Jacobson, and James Rumbaugh (History of UML, 2008). The predominant reason for developing UML was to provide programmers with a graphically-based common modeling language that could be used to visualize, specify, construct and document the artifacts of a software system (Kobryn, 1999). UML can be thought of as a much more elaborate type of software flowcharting. It incorporates more symbols that flow charts and diagram types that have contextual
meaning based on what is being described in a system. Software design patterns are solutions to recurring problems designed to increase benefits related to code reuse, code quality, code maintainability, and most importantly code re-usability (Aversano et. al., 2007). Simply stated, design patterns try to apply work already done by others to current coding problems. Gamma (1994) states: “Each design pattern lets some aspect of system structure vary independently of other aspects, thereby making a system more robust to a particular kind of change”. Using design patterns for developing DotAGWA provides a methodology for possibly re-using work done by others without re-coding the same application functionality. The goal of incorporating UML and software design patterns was to address shortcomings in an older technique of software design called flow charts. By using the combination of UML and software design patterns a much more robust system design was developed. However, using both tools could run the risk of information overload by providing too much detail that might eventually overwhelm system stakeholders. Consequently, the judicious use of both methodologies was applied throughout project development.

DotAGWA was designed as an internet-based application used to develop watershed management scenarios by allowing users to interact with spatial data through a web interface. The scenarios are processed using a hydrologic model. Output is delivered and presented to the user for his/her review and further analysis. The goal of this dissertation has been to present the three major phases in developing the DotAGWA application. The three manuscripts reflect three important phases of the project. The first phase was designing the application, the second phase was implementation, and the third involved
findings that emerged through work on the application itself and through literature review used to support both the application and this dissertation.

**History of AGWA**

Simply stated, the purpose of AGWA is to facilitate the parameterization and execution of hydrologic models using a map-based interface. An AGWA user specifies an area of interest or study area within a spatial data set. Using the study area the user develops a land use or land management scenario. The user delineates the study area’s watershed, selects soils and land cover input data and runs a hydrologic model for the user-defined inputs. After running the hydrologic model the output is presented to the user either graphically on the map interface or in plain text format.

AGWA was originally developed by Scott Miller, Darius Semmens, and Ryan Miller as part of their respective PhD research projects (Miller et al., 2007). The original version of AGWA (AGWA 1.0) ran on Microsoft Windows using Environmental Systems Research Institute’s (ESRI) ArcView 3.2 GIS (ESRI, 2001) application and was programmed using ArcView’s Avenue scripting language. The original version of AGWA run as a customized ArcView application therefore if the user did not have ArcView they could not use AGWA.

The current version of AGWA (AGWA 2.0) has been primarily developed by Shea Burns as part of his master’s thesis research (personal communication, January 2008). AGWA 2.0 is currently being programmed in Microsoft Visual Studio’s VB.Net programming
language and runs under ESRI ArcGIS/ArcMap desktop GIS application. AGWA 2.0 has the same features and functionalities as the original version as well as a few additional components that have been added at the recommendation of the user community and project stakeholders.

**History of DotAGWA**

The goal of DotAGWA is to provide all or most of the functionality described in AGWA through an Internet-based web application. DotAGWA brings the same GIS tools and hydrologic models to people who may or may not have access to the software required in AGWA. A secondary goal that is common to both AGWA and DotAGWA is the minimization of time users have to invest in learning both GIS software and hydrologic models. The first version of DotAGWA was developed by Ryan Miller as part of his dissertation research (Miller 2004). DotAGWA was originally developed using a combination of Java and Visual Basic. The Java side of the application ran most of the interface functions and the Visual Basic components handled the geo-processing and hydrologic model execution. The current version of DotAGWA was developed using Microsoft’s Visual Studio.Net and the C# (C-Sharp) programming language. Because the current version uses Microsoft’s Visual Studio, DotAGWA is able to re-use the existing geo-processing functions found in AGWA 2.0 without having to re-write the same code.

The development of DotAGWA occurred in two phases. The first phase was developing a software design document using UML. The design document serves as the
application’s blueprint. It is also a living document that is intended to capture changes that needed to be addressed as the application progressed. The second phase of the project was the development and implementation of the DotAGWA prototype. The test data for the prototype were derived from geospatial data of the Walnut Gulch Experimental Watershed (WGEW) located in Tombstone Arizona. The Southwest Watershed Research Center (SWRC) has developed spatial data for the Walnut Gulch Experimental Watershed. The SWRC is part of the United States Department of Agriculture’s Agricultural Research Service. WGEW is one of the most densely gaged and monitored watersheds the world. The prototype also uses web services that were developed for running the hydrologic models. The web services were written in the Java programming language under Ryan Miller’s original research and were re-used to save application development time. DotAGWA is also dependent on ESRI’s ArcGIS Server, which is used to display and manipulate spatial data (ESRI, 2004). Microsoft’s Internet Information Server (IIS) is the supporting web server software used to run DotAGWA as an Internet application. Application development was completed using Microsoft’s Visual Studio 2003 and the C# programming language. DotAGWA is dependent on a spatial database for storing static spatial data and spatial data created by user interactions with the application. A personal geodatabase, which is a proprietary database format developed by ESRI, was used to store and serve this spatial data. Finally, most of the spatial data used in DotAGWA had to be pre-processed which was completed using ESRI’s ArcGIS/ArcMap application (version 9.2) and ESRI’s Spatial Analyst extension.
The design section of the manuscript (Appendix A) develops a case study demonstrating software design techniques and procedures used to develop the application. The implementation section of the manuscript (Appendix B) shows how, after application design, the system was implemented and what was discovered while building the system and how those discoveries would then influence the design documents. Finally, Appendix C shows how similar work in this area lacks discussion on design procedures. The research articles tend not to discuss any design procedures and this may be creating a significant knowledge loss or gap and with the current demand for standardizing the procedures on building applications like DotAGWA any knowledge gaps could be detrimental to the outcome or value of the project.
PRESENT STUDY

The research in this dissertation will be presented in three manuscripts. The manuscripts are framed as documents in publication-ready format covering the following three research objectives.

Overview of the Three Papers

FIRST MANUSCRIPT: Design Document – This paper documents the design process used in developing DotAGWA and discusses and analyzes findings discovered during designing the application. Among the key findings was how useful UML models can be in communicating about components and processes of a software application.

SECOND MANUSCRIPT: Case Study – This paper documents a case study, which demonstrates an implementation of the DotAGWA application and includes findings on complexity issues and logistics that necessitated our altering the design of the application.

THIRD MANUSCRIPT: Findings – This paper demonstrates findings related to UML, software design patterns and stakeholder communications discovered during the design and implementation phases of the application. This paper will also present an analysis of how implementing software design methodologies like UML and design patterns may not be solely useful during the software design phase, but may have notable beneficial influence on the research and analysis of research projects involving building links between GIS applications and hydrologic models.

As a group, these papers provide a thorough examination of the processes involved in constructing a web-based application that links hydrologic models and GIS applications.
The overall objective of this dissertation is to present a framework for developing this type of web-based application and to provide useful information that may be used by researchers, GIS analysts or application programmers.

The research associated with the DotAGWA application development has revealed many complex issues that exist when trying to develop computer applications that link GIS applications to hydrologic models. The complexities tend to focus on the following important areas:

- The complexity of the hydrologic model and how the model describes physical watershed components
- The structure of the supporting input data sets being used to parameterize the models
- The complexity of the technological resources being used to construct an integrated system
- End-user perceptions and requirements of an integrated system

Further research on any one of these complexity issues will lead to a better understanding of systems that integrate hydrologic models and GIS software as well as provide insight into how GIS applications and hydrologic models might be developed in the future so that the complexity involved in building the links between the two domains may be facilitated.
As of the writing of this document, the second manuscript was accepted for publication in the conference proceedings of the 2007 Summer Computer Simulation Conference (SCSC) sponsored by the Society for Modeling and Simulation International (Cate, 2007).
REFERENCES


APPENDIX A MANUSCRIPT 1: DESIGN OF THE INTERNET BASED AUTOMATED GEOSPATIAL WATERSHED ASSESSMENT (AGWA) TOOL, DOTAGWA

To be submitted to *Ecological Modeling and Software.*

Expected date of submission Fall 2008
ABSTRACT

Land and water-resource management issues have changed from local or small scale problems to regional, spatially complex problems. These problems require interactions between diverse stakeholder groups. The technologies used to manage or understand these problems have become increasingly complex over time. The demands placed on the technologies have also increased in ways that require the integration of both new and old technologies many previously considered incompatible. For example, there is a growing need to integrate existing, scientifically validated hydrologic models with geospatially enhanced software applications in an effort to facilitate the parameterization and execution of these hydrologic models. The goal of this paper is to outline a methodology for such a project and discuss the technical details in the software design of the resulting application created in this project and how the design aspect plays a key role in the overall success of the project. The paper will discuss how using the Unified Modeling Language (UML), Test Driven Development (TDD), and Unit Testing help ensure the success of the application and maximize the application’s longevity. Finally, because integrating hydrologic models with geospatial data is becoming more important and since little research has been documented in this field, this paper will address an application development framework based on UML and TDD that offers a standardized solution. The benefit for having such a framework is to promote the construction of an application that is flexible and extensible, where other hydrologic or environmental
models or new geo-processing functions can be added without causing major application code changes.
INTRODUCTION

The United States Department of Agriculture (USDA)-Agricultural Research Service (ARS) and the Southwest Watershed Research Center (SWRC), in cooperation with the U.S. Environmental Protection Agency (EPA)-Office of Research and Development (ORD), and the University of Arizona (UA) have developed a geospatially enabled software application for parameterizing and executing hydrologic models. This Geographic Information System (GIS) tool, known as the Automated Geospatial Watershed Assessment (AGWA) application (Miller et al., 2007) is meant to facilitate the distributed hydrological modeling process as outlined in the EPA Research Plan by Goodrich et al., (2000 - EPA/600-00/042). There are two versions of AGWA, the desktop version (known simply as AGWA) and a web version (known as DotAGWA).

As a GIS application AGWA provides users the ability to define a watershed based on a digital elevation model (DEM). Within the area of interest users can develop land management scenarios. The scenarios include land cover/use type(s), soil type(s), and climate conditions. After developing a scenario, also known as a management plan, the user can direct AGWA to configure the watershed input parameters for a hydrologic model connected to the application. AGWA will use the input data sets to run the user-selected model and display the output to the user both in graphical map-based format as well as in tabular format. A key feature of AGWA is its ability to let the user re-run a model with variations in management plans in order to create management scenarios.
AGWA is a complex application requiring the integration of geo-spatial data, tabular data, relational database management systems, GIS software, and hydrologic models. The complexity of this type of system requires a thorough understanding of how all the different components are related to and interact with each other. Figure 1 describes the major processes and data sets that AGWA uses in creating watershed analyses. The application user defines a watershed by identifying one or more watershed outlets. The outlet points are used to create one or more watershed outlines. AGWA uses the watershed outlines to subdivide the watershed into model elements. The watershed elements are defined by the user selected hydrologic model (SWAT or KINEROS). Next, the application allows the user to select soils and land cover data that are intersected with the watershed’s spatial data. Typically, the soils and land cover data have associated lookup tables that are used to obtain the necessary soils and land cover types. This phase is called parameterization. Following parameterization, the user can select pre-defined rainfall data or create user-defined precipitation data. In either case the rainfall data is formatted to fit the respective model’s precipitation input requirements. Finally, the user-selected model is run using the developed input data sets and model output is displayed in the map component of the interface.
Figure 1 AGWA System Navigation

Navigating Through AGWA

- Generate Watershed Outline
- Subdivide Watershed Into Model Elements
  - SWAT
  - KINEROS2
- Intersect Soils and Land Cover
- Choose the model to run
  - Daily rainfall from...
    - gauge locations
    - Thiessen map
  - Generate rainfall data
- Storm event from...
  - NOAA Atlas 2
  - pre-defined return-periods
  - user-defined
- Run the Hydrologic Model & Import Results to AGWA
- Display Simulation Results
  - SWAT output
    - evapotranspiration
    - percolation
    - runoff, water yield
    - transmission loss
    - sediment yield
  - KINEROS output
    - runoff
    - sediment yield
    - infiltration
    - Peak runoff rate
    - peak sediment discharge
  - Visualization for each model element

Grid
Polygon
Look-up tables
External to AGWA
Based on the model shown in Figure 1 our research team’s creation of UML models and employment of TDD techniques has created a development environment that has facilitated visualizing the application components, visualizing linkages between components, and testing the components and linkages.
METHODS

System Design

Developing a model of the tool itself, the AGWA system, is vital for building a knowledge base that can be used to ensure all the pieces of the system are connected and that they are connected in the appropriate places and passing the correct information. The methodology we chose for modeling the AGWA system is the Unified Modeling Language (UML). UML is a language that can be used to describe the components and processes of a complex system. It is a language that uses graphical representations (e.g., circles, squares) to represent the components of a system and connectors between the components to represent communication between components and/or flow from one component to another. Since UML is a language it can be used to model systems from many different domains. Companies offering information technology services use UML to represent to themselves the ways their IT services are (or are not) reaching the public or their clients. A variety of types of businesses use UML to model the flow of data and the content of their data intra-organizationally (Kimmel, 2005). UML is a flexible tool that also allows it to be extended to fit many domains. Most importantly, UML models can unambiguously define system objects (Balram, 2006). The AGWA/DotAGWA UML model was created using Microsoft’s Visio (Microsoft Corporation, 2007) and Sparx System’s Enterprise Architect (Sparx Systems, 2008). Both applications allow users to develop UML models using a graphical interface.
User and System Requirements

In the case of the AGWA project, there are of course users as well as the system (AGWA) itself. User and system needs can be modeled by UML via the application of use case diagrams. Use case diagrams allow us to build a visual representation of how the system will be used. The use case diagram allows us to express the essential functions of a system visually. Use case diagrams do not show the order of operations; however they do provide an excellent mechanism for determining application requirements and modifying those requirements over time. The use cases modeled in this paper include three base level diagrams that are included to show fundamental functions required in the application. Two use cases are also included that model the two most important features in the application. The first is the watershed delineation use case and the second is the watershed discretization use case. These two features of delineation and discretization are the foundation for data sets developed when working with an AGWA project.

Primary Use Cases

Figure 2 shows the primary use cases in AGWA. Users can log in, develop a management plan, manage their account, and/or review and manage existing management plans. These are the base level use-cases and each one of these can be broken down into greater detail. The “includes” connectors in this diagram also show the use cases that are dependent on the user logging into the system. The box surrounding the diagram represents the system boundary, which in this case can be
viewed as the separation between the user’s internet browser interface and the servers hosting the application.

**Figure 2 AGWA Primary Use Cases**

![Diagram showing user interactions with AGWA](Diagram)

**Secondary Use Cases**

**Plan Management Use Cases**

The primary use-cases in Figure 2 can be further sub-divided into use cases providing greater detail describing the interactions between components and the user.
Figure 3 depicts the break down of use cases for plan development. The use cases in plan development identify six functions the application implements in helping a user develop a management plan. The user delineates a watershed for one or more outlet points. The delineation(s) is discretized and parameterized for soils data and land cover data. Finally, the user can execute a hydrologic model using the data developed during the previous processes. The use cases demonstrated so far are just a small number that have been developed for the application.

Figure 4 demonstrates functions that allow users to interact with existing management plans. In this case the user can view an existing plan, load an existing plan, delete a plan and/or modify a plan.
Watershed Delineation Use Cases

Watershed delineation is the root process when working with AGWA. All management plans are based on watershed delineations. Watershed delineations are based on one or more user selected outlet points as well as underlying input data sets not represented in the use case diagram. The delineation use case does show the operations and objects required for the delineation. The delineation use case has retrieval methods of outlet data and flow direction data, and a processing method used to delineate the watershed(s).

Figure 5 does not show all of the details required for watershed delineation. The details behind the inputs in use cases are typically modeled in state diagrams and class diagrams, which are discussed later in this paper.
Watershed Discretization Use Cases

The last of the three core functions in AGWA is the watershed discretization function on (Figure 7). The discretization process requires a delineated watershed, a flow direction raster data set, a flow accumulation raster data set, a digital elevation model, an outlets spatial data set, and a model type. Using the inputs, the watershed is subdivided by streams and given these inputs AGWA will subdivide the watershed into streams and elements as defined by the chosen hydrologic model. For example, in the case of the KINEROS model, a delineated watershed is subdivided by the stream network for that
watershed. This produces a geospatial feature set that is categorized as a set of planes (Figure 6) (Miller et al., 2007).

Each plane is related to a stream reach. The geospatial representation of the watershed has been formatted into a template that is a compatible representation of KINEROS model input. Subsequent processes in the AGWA application populate the input template with the necessary data (i.e., soils, vegetation) which are then used to run the model. Again, it is important to remember that use case diagrams represent functions or features available in the system and do not represent the order the functions are accessed.
The use cases discussed here provide meaningful information about what the application can and should do. The number of use-cases demonstrated in this paper was kept to a minimum because the goal of this paper is not to provide a complete UML model of the AGWA application, but instead a description of how and where UML was used as part of the AGWA application design process. Finally, use case diagrams do not define system interaction. However, UML uses interaction diagrams to model behavior and interaction between objects in a system.

**Activity Diagrams**

Modeling system use cases describes the core functions required in DotAGWA. Activity diagrams extend use cases to describe the sequence of actions in each use case (Kimmel, 2005). An activity diagram uses simple symbols and plain text to describe the processes.
in a use case. Consequently, activity diagrams can be understood by all project stakeholders. Conceptually, activity diagrams are similar to flow charts. Flow charts typically model process interaction by symbolizing starting and stopping points in program execution, data flows, decisions/branching, and flow paths.

**Login Activity**

Figure 8 is a simple activity diagram showing processes involved in the login use case. The user accesses the application through a web browser and is presented with the login screen. The user provides the required login information and submits the login information to the system.
The system checks the user’s identification and password against a user data table and either validates or invalidates the attempted login. If the user is a valid user then the system determines the user type (i.e. general or administrator) and directs the client application to present the appropriate start page. If the attempted login is by an invalid user then the system redirects the application to the login page. This is a simple example of an activity diagram in the application. The more complex diagrams are described in the delineation, discretization, and parameterization activity diagrams.
Delineation Activity

The delineation process is the starting point in the AGWA application for building a management plan (Figure 9).

Figure 9 Delineation Activity

The goal of this process is to allow the user to create one or more outlet points in the study area. The outlet points, which need to be located on the drainage network,
determined by the flow direction input, are used to create one or more watershed
delineations. The delineation process relies on the ESRI RasterHydrologyOperator. This
is a class or function in the application programmer’s interface (API) that has hydrologic
geo-processing functions. In this activity diagram the watershed method of the
RasterHydrologyOperator is used to create the delineations. The watershed method
requires outlets, in raster format, and flow direct raster data as inputs. The method
produces a raster data set as output, representing delineated watershed(s).

**Discretization Activity**

Discretization uses the watershed delineation and a watershed catchments data set to
develop a spatial data set that describes the drainage network (Figure 10)
The drainage network includes streams and catchments that feed into each stream. The discretizer requires a hydrologic model type as input (e.g., KINEROS) that is used to configure the drainage network that is appropriate for the hydrologic model.
Land Cover Parameterization Activity

Land cover characteristics and the discretization are inputs in this activity diagram (Figure 11).
The discretization spatial data and land cover databases are intersected. The resulting intersection produces land cover data for the region contained by the discretization. The resulting land cover data are formatted and matched to the discretization data set and
ultimately used as input to the user-selected hydrologic model. The application is flexible enough to allow the user to select different land cover types.

Soils Parameterization Activity

Soils characteristics input are captured by intersecting the delineation and the input soils type (Figure 12).
The soils type is determined by the user. Typically, the application provides soils data from the State Soil Geographic database (STATSGO) or the Soil Survey Geographic database (SSURGO). The intersection of the discretization data set and soils data set adds soils attribute information to the discretization. The new soils information is used as input parameters to the user-selected hydrologic model.
Land Cover Parameterization

Land cover parameterization is processed similar to the soils parameterization process (Figure 13).

Figure 13 Land Cover Parameterization
DotAGWA intersects the discretized watersheds with land cover data and populates the land cover attributes with values found for the user-selected land cover type. Like the soils data, the new land cover attribute data are used for the hydrologic model’s input.

**Model Run Activity**

The model run begins by writing all input data collected in the discretization process and writing that data to input files that fit the selected hydrologic model’s input format (Figure 14).
Next, the user-selected model (executable file) is copied to the same location as the input files. After the application copies the model, the model is run. The model produces output, which is read and stored for use by other application components.

**View Output Activity**

View output activity is a simple activity diagram (Figure 15).
The system at this point has a delineated watershed which has been discretized and parameterized for a user-selected hydrologic model, soils type, and land cover type. The hydrologic model has been run and its output stored. Now the system can display the output in a user specified output format (e.g., PDF, spreadsheet, text file).
System Interaction Diagrams

Interaction diagrams can be developed using the objects defined in use case diagrams. Typically, there are two methods for modeling interaction in a UML model (Kimmel, 2005). The first method uses sequence diagrams and the second method employs collaboration diagrams. Sequence diagrams show objects and messages passed between objects horizontally along the top of the diagram. Collaboration diagrams use the same objects and messages, but arrange them in a spatial display (Kimmel, 2005). This paper and the complete AGWA design document uses sequence diagrams because with sequence diagrams it is easier to follow the communication paths and steps between objects as well as throughout the model.

Sequence diagrams define methods and data passed between objects identified in the use case diagrams. The resulting UML model is similar to flow charts.

Delineation Sequence

The sequence diagram shows the sequential progression of method calls in the delineation process including method parameters and parameter data types (Figure 16).
The y-axis represents time in the downward direction and the x-axis shows the objects participating in the interaction (Jia, 2002). In the case of watershed delineation, the user determines one or more outlet points. The outlets are passed to the delineator object.

The delineator object then requests the flow direction raster data set from the application’s spatial database. After receiving the flow direction data set, the delineator creates a raster hydrology operator. The operator uses the flow direction and outlets data sets as input to the watershed function. The watershed method creates and returns a watershed delineation as a geo-data set.
Discretization Sequence

Discretization requires a delineated watershed and a hydrologic model type as input (Figure 17).

Figure 17 Discretization Sequence

The discretizor object accepts the two inputs and retrieves a third input from a supporting database. The three inputs are used to format the delineation data to match the input format required by the hydrologic model. The resulting data template is able to store data specific to the hydrologic model.
Soils Parameterization Sequence

Developing the soils input data requires the discretization, user-determined soils type, and user-determined hydrologic model type (Figure 18).

Figure 18 Soils Parameterization Sequence

The discretization data are augmented with soils data defined by or contained within each sub-catchment in the discretization. The database used for soils data is determined by the user (e.g., STATSGO, SSURGO) and the soils data are formatted to fit the user-selected hydrologic model. The parameterization notifies the calling methods whether the process
is successful allowing the calling method to retrieve the soils data from the discretization data set.

**Land Cover Parameterization Sequence**

Developing the land cover input data requires the discretization, user-determined land cover type, and user-determined hydrologic model type (Figure 19).

**Figure 19 Land Cover Parameterization Sequence**

The discretization data are intersected with land cover data. The database used for land cover data is determined by the user (e.g., NALC) and the land cover data are formatted to fit the user-selected hydrologic model. The parameterization notifies the calling methods whether the process is successful, allowing the calling method to access the land cover data in the discretization data set.
**Run Model Sequence**

The hydrologic model can be executed after the program has completed the discretization process (Figure 20).

**Figure 20 Run Model Sequence**

Running a model requires the parameterization data and model type. The model controller creates a temporary output workspace, stores the input data set for the particular model, retrieves or locates the user-specified model executable file and runs the model. After the model completes it run, the model controller reads the models output, which is typically stored on the local file system and passes that to the main application as text. The text can be formatted and viewed by the user.
View Output Sequence

The application displays output created by the user by reading the hydrologic model’s output, which is typically in text format and could be stored in memory or on the file system (Figure 21).

Figure 21 View Output Sequence

The output is formatted to comply with the user-specified output format (i.e., PDF, spreadsheet, text file) and displayed via the client interface.
DISCUSSION

Like a building’s architectural blue print, the UML model provides the starting point for constructing the application. With the model, software engineers and analysts have a frame of reference for what modules (classes) should describe and what methods (functions) should do. The model also presents a visual reference describing how the application components interact and communicate. Consequently, software engineers can review proposed changes by referring to the UML model before implementing any changes. This provides a powerful tool for anticipating the consequences of a proposed change.

Prototyping Phase

Prototyping is essential in many areas of software engineering (Goma, 1983). It is considered an expensive process during the design phase of an application. It is however considered significantly cheaper than fixing a system after the system is in production (Goma, 1983). Application development began with building prototypes of use cases and methods. Software development standards recommend prototyping components in order to construct what-if scenarios.

Prototyping in this project involved constructing components of the application as small manageable test units. These units were run against input data sets. The input data sets emulated those created by an application user. Test cases were constructed to verify the inputs and outputs of each prototype. Prototypes were constructed that connected other
prototyped components in order to test and monitor component interaction and result accuracy.

**Unit Testing**

Unit testing is a component of test driven development (TDD) allowing the programmer to test modularized components of the system. In conjunction with TDD, unit testing attempts to improve the maintainability and reusability of object-oriented systems (Kulesza et al., 2005). Unit testing is a development tool, which in the case of DotAGWA, required programming modules containing code designed to test the functional components of the application before they were integrated into the system. Unit testing required the programmer to write test code anticipating the correct or expected output values. Unit tests fail when component tests fail which provides a useful mechanism for monitoring when, where, and why a component might fail. In the case of DotAGWA, unit testing was not a distinct phase, but a sub-phase of both prototyping and implementation. Before a prototype or implemented component was developed, a unit test was coded and checked to verify the component passed functionality tests. Most of the time prototype unit tests could be re-used in the implementation phase of code development so most tests did not need to be written twice.

Unit tests were developed using the NUnit testing framework (NUnit, 2008). NUnit is a testing framework that can be used in all Microsoft .Net programming languages. NUnit allows programmers to develop modularized tests that can be used to test individual application components (e.g., classes) or suites of components in order to test the
interaction of multiple classes. In the case of this application, unit tests were created to
test basic features of the core geo-processing functions including watershed delineation,
discretization, parameterization and model execution. In each case static inputs were
developed and used to test each component. Static inputs made it possible to test for
expected outputs as well as tests for execution correctness. As testing advanced the input
data sets evolved in complexity so they might emulate real world usage scenarios. For
example, a basic delineation test involved the delineation of a single outlet point. The
more advanced tests for this case used multiple outlet points to emulate a user delineating
more than one location in the project area. Another case of developing more detailed test
input data sets arose by emulating a scenario where the application user might delineate
for multiple outlet points and select the KINEROS model to run versus the SWAT
(Arnold et. al., 1998) model. In this case two static input test sets had to be developed
and varied from each other by setting the hydrologic model to run as SWAT or
KINEROS. The resulting unit test(s) confirmed the correct model ran and the expected
output was created. After all unit tests were developed and had demonstrated that the
geo-processing components of the application were producing the expected outputs, the
geo-processing components were then connected to the client interfaces that serve as the
entry point(s) into the application.

Finally, a system constructed by using test driven development is easier to validate in
terms of the data produces. Unit test suites can be constructed that produce output that
might be expected in real-world usage scenarios. The output from a test suite can be verified against real world data, which then helps validate the entire application.

**Implementation**

Application development began after the completion of the UML design document. However, before writing code a development environment had to be constructed that would allow us to write application code and release versions of the application as more features were added. First, the geospatial data set used to develop and test the application was downloaded from the Southwest Watershed Research Center’s geospatial data base. The geospatial data consisted of a 10m digital elevation model (DEM) of the Walnut Gulch Experimental Watershed, shape files containing soils data, land cover data, and climate data in the form of precipitation files. AGWA application could begin once the geospatial data was retrieved. Application development was conducted using Microsoft Visual Studio 2003 and later in the project Microsoft Visual Studio 2005. DotAGWA development required retrieving the same Walnut Gulch Experimental Watershed input data sets and installing ESRI’s ArcGIS Server application. ArcGIS Server hosts and delivers spatial data set over the Internet. Once the spatial data was being served over the Internet the web application was developed in incremental phases. Each phase ended with a stable release of the application and the stable version of the application was installed on a server that made DotAGWA available over the Internet. This setup allowed application development to continue and new features to be added to the application without disrupting the deployed or released version. Consequently, any
additions that may not have worked properly could be fixed before being added to the deployed version of DotAGWA.

Unit testing began after establishing the development environment. Unit tests were guided by referring to the design documentation and were conducted by using the same Microsoft Visual Studio tools as stated earlier. System components were pieced together after components passed unit testing. Completion of all system components was followed by simple user tests. Due to project time constraints a full system unit test was not conducted.
CONCLUSIONS

Incorporating TDD and UML design adds create more operational overhead for software development projects. If programmers and stakeholders are not already trained in UML and TDD then more time is required for learning UML and TDD. However, this cost can be significantly offset and is quite likely recovered by the improved sustainability of the application as well as its improved lifespan. Costs are recovered in areas like providing a well defined application scope to the programmers, which can keep programmers focused on the most important project tasks (Crispin, 2006). A third area is how TDD can facilitate communication between programmers and project stakeholders (Crispin, 2006). By establishing patterns where programmers write small-functional bits of code, test them, then verify with stakeholders all parties tend to keep communication channels open (Crispin, 2006).

In the case of geo-processing intensive application development, the test-driven development process (TDD) has proven to be quite beneficial. Prototype versions of the web application demonstrated that development took much longer when programmers didn’t follow TDD. This was because programmers where writing code changes that then took significant amounts of time just to wait for the whole application to compile, start, and execute changes that tended to be small. By implementing TDD development programmers are able to make code changes of whatever size, compile the modified objects and test the changes against representative input data sets without having to start the entire client interface.
A second benefit found by using TDD emerged during debugging processes. Similar to the previously mentioned benefit, debugging could be completed by tracing through the objects where new code changes occurred without having to run the application’s interface. This saved in debugging time and confusion by making it easier for programmers to track the location of problems in code by having many fewer application layers to trace through. TDD made it easier to see errors or modifications in the design of the software that were not visible until application components were constructed and running. This ultimately improved the ability of the application to flex to future program changes and enhancements. Finally, using TDD techniques tends to entice programmers to develop smaller more-highly reliable modules that are easier to understand and repair if necessary (Jansen et al., 2008).

Finally, employing the combination of test driven development and UML modeling created software development documentation that allowed project stakeholders to easily review and modify the design document(s). The stakeholders’ requested modifications were also easier for programmers to address, understand, and implement. The UML document(s) also provided a common language for technical (i.e. programmers) and non-technical (i.e. watershed managers) actors, facilitating communication between the two parties.
LITERATURE CITED


APPENDIX B MANUSCRIPT 2: DOTAGWA: A CASE STUDY IN INTERNET BASED ARCHITECTURES FOR CONNECTING SURFACE WATER MODELS TO SPATIALLY ENABLED WEB APPLICATIONS

Submitted, reviewed, and accepted to 2007 Summer Computer Simulation Conference:

ABSTRACT

The Automated Geospatial Watershed Assessment (AGWA) tool is a desktop application that uses widely available standardized spatial datasets to derive inputs for multi-scale hydrologic models (Miller et al., 2007). The required data sets include topography (DEM data), soils, climate, and land-cover data. These data are used to develop input parameter files for two USDA-ARS watershed runoff and erosion models: the Kinematic Runoff and Erosion Model (KINEROS2, Smith et al., 1995) and the Soil and Water Assessment Tool (Arnold et al., 2008). AGWA has proved to be a useful tool for many different applications. Not all potential users, however, have access to the geospatial data or software required to process it and run tools like AGWA. In addition, some potential users recognize the value in being able to use the application, but do not have adequate technical training to gather and process the necessary data and run the application through a geospatial information system (GIS) software platform. A Web-based version of AGWA, DotAGWA, was developed to address these issues and cater to a wider potential user audience. This paper describes the design and structure of the DotAGWA application and discusses important findings related to issues and problems that emerged during application development. In particular, important issues emerged related to configuring a system that would connect surface water models, originally intended as desktop applications, to a spatially enabled web application. Some of these issues include input and output file management for model runs when models are executed from the web-client to the server’s operating system, configuring the systems spatial and non-
spatial data requirements in a web server environment, and designing an extensible or at least reusable system architecture.
INTRODUCTION

AGWA, originally released in 2002, was developed using the Environmental Systems Research Institute’s (ESRI) ArcView 3.x and its Avenue scripting language. An updated version for ESRI’s ArcGIS 9.x platform, AGWA 2.0 is being released in 2007.

DotAGWA was designed in concert with the AGWA 2.0 desktop version and these two versions share the core GIS functionality. The goal of this paper is to provide information documenting the DotAGWA development process and findings that were and are being discovered as the application develops. DotAGWA is an Internet-based application connecting hydrologic models to a map-based interface. Users can develop management scenarios or define land use characteristics which are combined with supporting data sources to parameterize and run the hydrologic models connected to the application.

Applications similar to DotAGWA do exist. The Spatial Decision Support System for Watershed Management developed at Purdue University has watershed processing features similar to those used in DotAGWA. The Purdue System also allows users to run hydrologic impact and sedimentation models for a delineated watershed. Researchers at Purdue University have also developed an online web application that allows users to run the Watershed Erosion Prediction Project (WEPP, http://milford.nserl.purdue.edu/wepp/weppV1.html). WEPP is a simulation that estimates erosion using climate, land management, soils, and topography as inputs. Once inputs are defined through web-based input forms the user can run the model and download the output. However, the Purdue application does not allow the user to develop input data by
letting the user define a project area through a map-based interface. DotAGWA has unique features that facilitate interaction with hydrologic simulation models. By obviating the need for data compilation and processing and enforcing a controlled process for user interaction with the model, DotAGWA creates an environment where people with little or no training in hydrologic modeling and GIS can use this powerful technology for practical decision support. Via a spatially enabled interface, users can develop management scenarios, parameterize and run the hydrologic models, and compare model results. The features available in DotAGWA are also designed to help users share and visualize data. Groups can interact with the application to facilitate the communication and decision-making processes. Users access the application through an Internet browser interface. Application servers connect the user-interface to database servers (both spatial and non-spatial). The application servers also connect the user and data to the hydrologic models. Users can define management scenarios, attach models to a plan and have the application parameterize and run the models for the defined management plan.
APPLICATION DESIGN

The early stages of the DotAGWA project required developing a design document using the Unified Markup Language (UML) and other software programming techniques (Cate et. al., 2005). Important reasons for writing a design document included providing a roadmap for developers with clear application requirements. In addition, the design document also served as a bridge between AGWA 2.0 and DotAGWA. AGWA 2.0 has existing functionality that was identified as being reusable in the DotAGWA application. Since re-use is an important tenet in software development, AGWA 2.0 became a foundation for DotAGWA, providing a significant portion of the geo-processing tools needed in the DotAGWA application.

Application Users

As with AGWA, DotAGWA’s end users were identified as resource planners, developers, watershed management professionals, and many other interested parties (Cate et al., 2005). However, the scenario for application usage for the users had different requirements. A web-based application allows the user community to access the application from any computer connected to the Internet. Users can work together in activities like defining spatially explicit best management practices (BMP), test the effectiveness of BMPs in areas of interest and share the results with others via the Internet.
System Requirements

A system like DotAGWA has many hardware, software, programming tool, and data requirements. The development of the design document and knowledge gained from development of AGWA provided a solid foundation of information that could be used for DotAGWA application development, but because DotAGWA is an Internet-based application many new issues needed to be addressed.

Software

Like most other web applications DotAGWA has many different software requirements. Programmers need an integrated development environment (IDE). Data used to support the development and deployed versions of the application need to be stored in a common, easily accessible database. Web-server software is needed to make the application accessible over the Internet. Finally, and most importantly in the case of DotAGWA, a spatial data server is required to make spatial data available to the application.

Programming Environment

DotAGWA was developed using Microsoft Visual Studio 2003 (Visual Studio). In this environment programmers can develop in different programming languages that are supported in Visual Studio. In the case of DotAGWA, C# was chosen as the primary programming language.
**Database Server**

The application stores and serves both spatial and non-spatial data. The spatial data include digital elevation models (DEM), shape files, and attribute data that describe features like soil types, land-cover information, and meteorological information. The non-spatial data include, user information (e.g., user id, email), session information (e.g., model name, model output), and parameter look-up tables. The current version of the application uses a personal geodatabase (Microsoft Access file) for storing all of this information.

**Web Server**

Microsoft’s Internet Information Services (IIS) was selected as the web application server. The main reason IIS was chosen was because it is the web server that Visual Studio works with. IIS also comes installed on most Windows operating system servers. Consequently using IIS required no extra cost, and imparted no limitations to the developers or end users.

**Spatial Data Server**

The Web Server also has an additional spatial data server that allows the application to display and manipulate spatial data required by the project. The spatial data server is ESRI’s ArcGIS Server version 9.1. ArcGIS Server was chosen because it allows programmers to build web-based applications that can both serve and manipulate spatial data. Earlier versions of DotAGWA used ESRI’s ArcIMS map server, which only displays spatial data through a web interface and required using ESRI’s ArcSDE
application programming interface (API) to manipulate data. ArcGIS Server provides tools to the programmer through its API that allows spatial data manipulation.

**Data**

Spatial data from the USDA-ARS Walnut Gulch Experimental Watershed was utilized in the testing and development releases of DotAGWA. Ultimately, DotAGWA can be implemented on a national or regional scale, but at present it requires pre-loading data for project areas on a case-by-case basis. The base data sets include a DEM (raster/grid), land-cover/land-use (raster/grid), flow direction (raster/grid), flow accumulation (raster/grid), watershed boundary (vector/shape file), sub-watersheds (vector/shape file), stream network (raster/grid), soil maps (vector/shape file), climate (database), and land-cover/landuse parameter lookup database tables.

**Hardware**

Intel Pentium PCs running Windows XP Professional are used for application development. Any Windows-based PC can be used as long as the computer is able to run Visual Studio and ArcGIS Server. The deployment server is currently running Windows XP Professional, but in mid-2007 a new PC running Windows 2003 Server is going to be used to host the application. Because the spatial data server software can consume a great deal of system memory all computers have a minimum of 2 gigabytes (GB) of memory and at least 100 GB hard disk storage. The current development configuration, where there are very few web users accessing the application, is adequate for the current version of DotAGWA. However, when DotAGWA is opened to the general public, it will
be hosted on the Windows 2003 server making it capable of supporting enterprise-level user requests.
APPLICATION DEVELOPMENT FINDINGS

Application Development

One of the most important requirements for the DotAGWA project was to re-use as many features as possible that were available from the AGWA 2.0 project (Cate et. al., 2005). AGWA 2.0 has functions that automate watershed delineation, watershed discretization, model parameterization, and model execution. These are the primary functions that DotAGWA relies on. These functions allow a user to delineate a watershed, discretize the delineated watershed into model elements, derive parameter inputs for the selected hydrologic model, and finally to run the model using the parameterized input data set. Figure 22 shows DotAGWA’s system architecture. The application is hosted by an Internet application server. The Internet application server presents the web interface to the client(s).
The web interface has a map component that allows the user to develop application input by interacting with a map as well as typical web-based input forms. The application server process input and handles interaction between a supporting spatial database server as well as an application data server. Separating the two server types makes it possible to swap servers with minimal impact to the entire application. The spatial data server performs in a supporting role. The two servers are currently not two separate computers. However, the system is flexible enough to allow the database servers to be two physically different systems.
Interface Development

One of the first steps in application development was to construct web interfaces that were similar to the AGWA 2.0 application and its user interfaces. Figure 23 presents DotAGWA’s basic work-flow model. The workflow model represents the user’s activities from login through delineation, adding management-plan features (optional), model selection, discretization, parameterization, model execution and result/output processing. Because the application focuses on providing a tool that allows people to build and compare management scenarios the system has to track individual users and management plans created by those users. A user logs in and an application session variable is instantiated and used for the entire time that the user interacts with the application. The session variable that describes the current user also stores data that tracks information related to the user’s current management plan and has functions that allow the user to manipulate that plan and/or previously created management plans.
To begin, the user is presented with a map of the project study area (Figure 24), in this case the Walnut Gulch Experimental Watershed (WGEW) in southeastern, Arizona. At
this point the user has initiated a new management plan that is ready to be developed for whatever management scenario the user would like to create.

**Figure 24 DotAGWA Main Interface**

![DotAGWA Main Interface](image)

Figure 25 *Error! Reference source not found.* presents an example of the user creating a delineation point and executing the watershed delineation, which is shown on the map figure as the grey hash-lined area.
After delineation (Figure 25) the user can select a model to run against the new watershed and select precipitation input for that model. The current version of the application allows for user-defined precipitation events (design storms) in KINEROS2, or a pre-loaded rainfall record for SWAT. The user also selects land-cover and soil maps to use for model parameterization. With all input parameters set, the model can be run. Running the model also executes the discretization process, during which the watershed is subdivided into model elements (sub-watersheds and stream reaches) based on the model type that is selected. The two hydrologic models that are currently incorporated in the application each require a different discretization procedure.
**Initializing Model Run**

After watershed delineation the interface lets the user select a hydrologic model to run, a soils type, a land cover type, and precipitation type (Figure 26). After the model has finished running the user can click a button on the interface that loads the model’s output file into a web page (Figure 27). The user can review the output and/or cut and paste it into another document for further analysis or use in a report. Also, the user can select to have the map display update to create a visual representation of an output parameter (i.e., peak flow).

**Figure 26 Initializing Model Run**
Finally, DotAGWA allows users to manage BMP plans or scenarios (Figure 28). A user can develop a delineation, discretization, and define spatial features that represent a BMP. The BMP is stored in a supporting database. A model can be selected and run against a user-defined BMP and that output can then be stored as part of the BMP. Each plan can be reviewed or deleted from the user’s profile. As of the writing of this paper, development is proceeding in adding functionality that lets a user compare (non-spatially)
BMPs. Comparing BMPs will require the watershed to be near identical and model output for each scenario should originate from the same model. The comparison will be simplified. It will compare some of the most common output parameters like peak flow and sediment yield.

**Figure 28 Plan Management**

Developing the application has revealed many issues that are worth noting. An earlier version of the application was Java based and relied on calling the AGWA 2.0 functions as web services. Although an interesting concept, this approach proved to be problematic.
AGWA 2.0 creates personal geodatabases (Microsoft Access databases) to store user output for each watershed. Every time a watershed delineation request was executed the delineated watershed was stored in a personal geodatabase. The ArcGIS server had to access the geodatabase in order to display the delineation in the web interface. This worked for the delineation process, but because the personal geodatabase, Microsoft Access, is not really designed to be a multi-user database accessed from an Internet application, subsequent calls to the personal geodatabase for discretization caused file locking and security issues with the IIS web application server. The solution to this involved porting the entire Java application to the Visual Studio C# language and connecting the AGWA 2.0 tools to DotAGWA as a dynamic link library (dll). Because AGWA 2.0 is accessed under the same IIS user account on the server no web-based security rules were broken and the output personal geodatabase could be accessed as needed.

**Data Integration**

Programming DotAGWA began with our using the same standard data set available to AGWA 2.0 desktop users and used in development of the AGWA 2.0 utilities. The base data set covers the ~150 km2 WGEW in Tombstone, AZ. When using AGWA 2.0 users are required to set up a project, which involves path names to input data sets like the DEM, land-cover grid, etc. In AGWA 2.0 each user and each project can have different base input data. However, because DotAGWA is a web-based application that could be accessed by multiple users, DotAGWA is currently not configured to let users define their own input data sets; all data must be pre-loaded by an administrator. Because the
The application is a multi-user web application; each installation is unique to the project study area. However, the application was developed with enough flexibility to allow administrators to develop a base data set that is compatible with both AGWA 2.0 and DotAGWA, deploy a new version of the web application by installing to an IIS enabled folder, and configuring the application to use the new data by changing only one configuration text file. DotAGWA setup requires pre-processing of the base DEM to create the flow-direction grid, flow-accumulation grid and the stream-network grid. The AGWA 2.0 desktop application allows the user to derive these data sets from the DEM. The pre-processing is required because developing the dependent raster data sets is computationally intensive and slow. The current version of the application also uses default database look-up tables deriving model input parameters from land-cover classes and soil types. AGWA 2.0 allows users to edit these tables if needed. An enhancement currently in progress allows users to delineate and discretize a watershed, and subsequently change the land cover classes or types within that watershed. This feature will allow users to define future land-cover changes to watersheds and run the models against these changes to make estimates about the impacts the changes might cause. As stated previously, precipitation data is handled by letting the user create his/her own design storm or by using preloaded daily rainfall data. These methods represent base case implementations for the two hydrologic models supported by DotAGWA. More technically sophisticated options will be added later, such as directly interfacing with the National Weather Service’s web services. Finally, the application needs a way to track the development of a BMP, and to track user interaction with the application. The
following data model shows how users and BMPs are accounted for in the supporting application database. Figure 29 shows the database table structures and relationships used to manage supporting application data. An application user has a unique identifier (id), first name (fname), last name (lname), password (pword) and email address (email). Each user can have zero or more management plans. A management plan consists of a unique identifier (id), a unique user id (user_id), a plan name (name), a plan description (description), creation date (created_on), the model name used on the plan (model), and the model output (output). Each plan has zero or more points that consist of a unique identifier (OBJECTID), spatial point object (SHAPE), a user identifier (user_id), a plan identifier (plan_id), and a point type (type). Each plan also has zero or more plan lines that consist of unique identifier (OBJECTID), spatial line object (SHAPE), a user identifier (user_id), a plan identifier (plan_id), and a line type (type). A plan can also have zero or more polygons, which have a unique identifier (OBJECTID), spatial polygon object (SHAPE), a user identifier (user_id), a plan identifier (plan_id), and a line type (type). Finally, each plan has a watershed delineation and a watershed discretization. A watershed is composed of a unique identifier (OBJECTID), spatial polygon object (SHAPE), a user identifier (user_id), a plan identifier (plan_id), and a point type (type) and a discretization is made up of a unique identifier (OBJECTID), multiple spatial polygon objects (SHAPE), a user identifier (user_id), a plan identifier (plan_id), and a point type (type).
Model View Controller Design Pattern

The MVC design pattern is useful for separating the presentation components of the system architecture from the data storage and processing components of the system. The reason this separation is useful is if one of the three components the Model, View, or Controller changes or needs to be replaced the alterations do not cause major changes in other parts of the system like the interface. In the case of DotAGWA, web forms and
pages serve as the view or user interfaces. Requests generated by user interaction with the view are handled by controllers with specialized functionality. For example, the application has a controller for handling interactions with the map and a second controller that handles interactions between user interfaces and the surface-water models. The processing, transfer and storage of data are handled by the model component, which serves as a bridge between the application and its supporting databases. The most important reason for using the MVC design pattern is to maximize cohesion and minimize coupling between application components. Striving for this minimizes the negative impacts to major changes in the software. If, for example, the application needed a completely new interface it could be developed and have almost no impact on the rest of the application. Or, if the GIS server software changes then minimal changes to the application code controlling communication between the map server and the view or model would be necessary.
FINDINGS AND SUMMARY

The DotAGWA project has involved integrating five major components: spatial databases; internet web servers; spatial data servers; relational databases; and standalone surface-water hydrology models. The overall goal has been creating an application that addresses the growing need to build tools that increase our knowledge and understanding of landscape management and change. The internet web server, the relational databases, and the spatial data servers are all designed to work together. Consequently, integrating these components has required the least amount of effort. There have been some issues related to connecting the web application server, the spatial data server, and the AGWA 2.0 geo-processing tools. AGWA 2.0 uses personal geodatabases to create and store new spatial data sets. Personal geodatabases are a proprietary data format created by ESRI, but they are basically Microsoft Access database files. As mentioned in section 3.1.3 (System Configuration), the AGWA 2 functions were originally accessed as web services. However, this proved problematic because AGWA 2.0 produces and uses a personal geodatabase for storage. Because the personal geodatabases are Microsoft Access databases they are not well suited for internet-based applications. The particular issues that arose were related to file locks on the database during geo-processing and scale issues related to the number of users accessing the web application.

Two things were done to address these issues. First, the original version of DotAGWA was re-programmed from Java to C# using Microsoft Visual Studio. Second, DotAGWA has special operating-system level user permissions and utility functions that let it access
personal geodatabases. In the very near future, the personal geodatabase usage in DotAGWA will be phased out and spatial data creation and storage will be handled by either Microsoft’s SQL Server or Oracle’s Enterprise database server, each with spatial data features enabled.

Connecting the surface-water models has proved to be one of the most challenging parts of the application. Each model has a unique way of defining a watershed and its sub-basins. Each of these basin definitions requires a unique parameterization method, which in turn requires application code to be created that transforms the spatial information into an input text file that the model can read. The surface water models are technologically older, created using FORTRAN and meant to run as standalone desktop applications. Consequently, a strategy for running these models in a shared web environment has been developed whereby a model wrapper object is created that reads and writes the input data for the model. Each model is executed in a unique file-system folder. The folder is created based on the current user ID and a time stamp. The model executable and input are stored to this temporary location, the model is run, and the model output is read from that location and stored in session variables in the web application. This method has so far been successful with one minor negative consequence: the need for removal of temporary files and folders.

Lastly, open-source GIS tools were considered during the design phase. DotAGWA uses time and resource intensive raster geo-processing tools. The open-source community has
a limited number of tools that can handle the type of geo-processing functions needed. However, the Federal agencies supporting DotAGWA development have negotiated licensing agreements providing access to ESRI’s geo-processing tools. This, together with extensive documentation and very large user community for ESRI’s products, the use of proprietary GIS products was a logical choice.

In conclusion, the goal of this paper has been to describe the methods and procedures used to construct a web-based application that can be used in decision making processes related to water resources, land-use / land-cover change and best management practices. With rapidly increasing demands on our natural resources and the potential for system-altering climate change, there is a heightened need for tools that exploit new technologies to make complex process models available to a wide audience, namely local governments, NGOs, and resource managers.
REFERENCES


APPENDIX C MANUSCRIPT 3: FINDINGS - EMERGENT PATTERNS IN DEVELOPING A WEB-BASED AUTOMATED GEOSPATIAL WATERSHED ASSESSMENT TOOL

Prepared for submission to Transactions in GIS proposed submission Fall 2008
ABSTRACT

With the increase in computing power and the decrease in costs for accessing that computing power the need for developing specialized applications linking GIS applications to stochastic and process models is growing. Supplementing this increase in demand is the world’s growing population and demands on natural resources. The demand on developing these applications has produced a new set of problems that are somewhat different than the research problems using these integrated applications. We are seeing a growing need for developing or extending the use of software engineering technologies, for extending applied linguistics, and the growing use of artificial intelligence knowledge bases to develop an ontology or unifying knowledge base that can be used to develop and discuss integrated GIS and process model applications. This paper addresses these topics and issues with regard to the development of the Internet-based Automated Geospatial Watershed Assessment (AGWA) tool called DotAGWA. The paper discusses issues that emerged during the development of the DotAGWA application that provide valuable information and an example ontology that could be used in other related research projects. This research highlights how software development tools like the Unified Modeling Language (UML) and software design patterns can serve as useful resources in the development and growth of a project such as DotAGWA that links GIS and hydrologic models.
INTRODUCTION

Today Watershed managers benefit from and are also sometimes hamstrung by the presence of the computer in their work environment. The development of Geospatial Information Systems (GIS) software, for example, has made the display and exploration of geographic data in watershed management much easier in comparison to times where only paper-based maps were available. Presently, a watershed manager can—in a manner of minutes—visit a spatial data provider’s website, download the spatial data for an area of interest, and collect in one place in an organized fashion the field-collected data on that area of interest—even manipulating those data and generating quantitative analyses of those data. Before the widespread availability of software such as GIS, to have performed comparable analysis would have taken hours, possibly days. In the past, participants in a comparable project would have had to spend hours pouring over maps, physically writing down information on the watershed’s attributes, and then conducting and presenting analyses of those data by hand. But one key aspect of these projects of past times bears a closer look: all participants to the project were speaking the same language. They could, using everyday language, meaningfully discuss additional tasks they would like to tackle, tasks to put on the back burner, goals that were feasible or non-feasible and new directions their project might take.

Today, a “digital divide” prevents this type of communication in watershed-based projects. Programming languages and software applications have proliferated. Multiple
programmers on a project team may not have a good mutual understanding of how their code will affect other programmers’ contributions (One may be a Java person, another, a C++ whiz.). Another project participant might program in neither of those languages but be allied with a particular software application, and thus only “speak GIS,” for example. Software-allied participants might wish for outcomes programmers know to be unrealistic and vice versa.

Some researchers on a project may not program in any of the languages in use on the project, whatsoever, but bring to bear strong domain knowledge for the project. These latter participants, in fact, may have initiated or be funding the project or even be directing the project—proposing initiatives for the project to address. The programmers would, in these cases, be attempting to translate non-programmer directors’ wishes into actual code. What has been requested might or might not actually be feasible— but without a lingua franca for communicating how or why a request cannot be met, project participants are likely to experience difficulty keeping each other appraised of progress or lack thereof in a project. Misunderstandings are common in these circumstances and almost certainly are a factor contributing to unknown problems encountered during the development of a solution and could be a major contributing factor to the failure of a project.

What is needed is a language that any project participants (layperson directors, software specialists, programmers and non-programmers alike) could speak to each other to
envision initiatives for their project that are feasible. With a standardized language, all parties to a project could communicate with each other about the natural resources domain-specific issues of their project, as well as the software engineering issues.

In addition to the need for a lingua franca, there also exists a need for maximum flexibility in today’s software design processes. Software engineering employs an engineering practice called design patterns (DP) to enhance the flexibility and lifetime of information system projects. A design pattern is a description of communicating objects and classes that are customized to solve a general design problem in a specific context. Each pattern represents a common and recurring design solution which can be applied over and over again in different problem-specific contexts (Ventura, 2008). Design patterns provide a resource for others to reuse and/or re-examine application code without having to re-invest time and resources to develop the code or data. A library of design patterns in geographic information science (GIS) would prove to be an invaluable resource to the research community. Using UML in conjunction with design patterns would prove to be two extremely powerful tools in enhancing and extending the language created to develop research in geographic information sciences.

This paper will focus on using an existing application, DotAGWA as the foundation for developing an example of the lingua franca. DotAGWA is the internet-based version of the desktop ArcGIS Automated Geospatial Watershed Assessment (AGWA) tool. AGWA was constructed as an open-sourced GIS-based system that integrates hydrologic
models and facilitates model parameterization, model execution, and output presentation. The application allows users to develop landscape management scenarios using hydrologic models with varying landscape scales (Miller et al., 2007). In an effort to create an efficient application, the development of DotAGWA was dependent on the geo-processing and hydrologic processing tools developed for AGWA. This paper will review the successes and failures of the methodology and demonstrate how the knowledge gained provides an excellent foundation and justification for developing a standardized language for designing and developing integrated GIS applications.
ISSUES

Ontology and Knowledge Bases

Key to developing a common language is establishing a robust knowledge base or ontology in Geographic Information Science. This in itself is critical and an emerging theme in geographic information science (Mark, 2002). However, the vastness of information in geographic information science may mean it is impossible to create a definitive knowledge base (Deliiska, 2007). What is possible though is the identification and study of a small scale project that can be used as a test case for developing a domain specific ontology and lingua franca.

For clarity it is important to understand the use of the words ontology and knowledge-base in this paper. Both words are used synonymously throughout this paper. The term ontology is used in information systems and philosophy in a variety of ways (Mark, 2002). Guriano and Giaretta (1995) also provide an informative discussion on using the terms. To summarize Guriano and Giaretta, suggest that “ontology” can have multiple interpretations and that each interpretation can be very different. They propose using the word “Ontology” (with a capital “o”) as the term meaning a philosophical discipline. They develop the word “ontology” (lower case “o”) as being related to knowledge bases that have the goal of sharing knowledge. However, the paper desires the word to not have a unique or precise meaning, but also offers the idea of having two levels of interpretation. The first use is conceptualization which denotes semantic structure that reflects a conceptual system. The second use is ontological theory which denotes
ontological knowledge a.k.a. knowledge bases. This paper will primarily rely on the knowledge base form of the definition of ontology.

**UML Modeling**

The Unified Modeling Language (UML) is a software engineering tool that seems promising for representing the knowledge base being developed in this research. UML displays programming commands graphically, allowing non-specialist users and onlookers, alike, to see and understand at the software level the possibilities for proposed natural resources project initiatives. As an example, Figure 30 shows a simple Use Case diagram. Use Case diagrams can are primarily used to document a system’s macro-level requirements. Alternatively, Use Cases can also be thought of as the list of system capabilities (Kimmel, 2005). The Use Case (Figure 30) shows actions that a web client can, and in some cases has to do in an application.
This example shows that a web client can log in to the application, load a management plan, develop and or modify a plan. Each use (e.g., login) is defined by asking end users what functions or needs the application has to provide. Developing the list of uses requires interaction between users and the programming team. Consequently, invested parties develop a solid foundation for communication that can lead to the successful completion of the project.
Why UML?

Admittedly, UML can have a steep learning curve (Balram, 2006). However, once users have a basic understanding of the language’s symbols, reading UML diagrams can ascertain information very quickly and quite likely make up for the time spent in having to learn UML. UML also helps minimize the ambiguity that can be created in spoken and written language. UML diagrams also mirror what we already do by using graphs to summarize data analysis. UML diagrams simply summarize what objects in a system should do and how they relate to each other.

UML in Geographic Information Science

Recently, we do see a few examples of UML in Geographic Information Science to convey components and/or activity within a system and/or system processes. However, discussion of the UML itself is typically limited and frequently non-existent. Generally, most papers depict system interaction through simple flow charts and system objects through over-simplified class object symbols. Although useful, flow charts may not show quite as much information as the equivalent UML diagram. Research articles lacking UML documents as part of the paper may not be conveying information that would be extremely useful to readers. Also, by using UML as a significant component in the research article it would be easier for readers to understand processes used during development of the research project. This may put research under greater or different types of scrutiny, but it also serves to strengthen any geographic information science’s
research project or any natural resources related project that involves computer programming.

**Design Patterns**

Design patterns originated as a way to re-use work done by other programmers. Patterns emerged as people began to see patterns in the types of problems being solved by computer programs. Patterns also help keep a system as flexible as possible. A design pattern is a description of communicating objects and classes that are customized to solve a general design problem in a specific context. Each pattern represents a common and recurring design solution which can be applied over and over again in different problem-specific contexts (Ventura, 2008). Software design patterns originated in the late 1980s. Ward Cunningham and Kent Beck created a set of patterns used in the development of user interfaces (Gamma, 1994). At about the same time, Jim Coplien developed a catalog of C++ patterns called idioms. Also, Erick Gamma recognized the value of recording the recurring design structures while working on his doctoral thesis (Rising, 1998). These and other people met and developed their discussion on design patterns at a series of Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA) workshops beginning around 1991 (Rising, 1998).

**Design Patterns in Geographic Information Science**

At the present time, discussion or use of design patterns in GIS literature is sparse. Geographic information science related projects use common data sets (e.g., digital elevation models, soils database, and land cover database). We see processes used
repeatedly between different GIS projects (i.e., watershed delineation, watershed discretization, watershed parameterization). These data sets and processes could and should be well defined so they can be reused by others, saving both time and resources. Lastly, constructing design patterns specifically applied to GIS and formalizing them in a peer reviewed repository makes these items defendable for scientific review. In a community agreed upon and standardized repository, GIS design patterns can be developed over time and verified by the whole community, leading to robust and validated research tools.

**GIS Software and Process Model Linkage**

GIS software has existed for many decades and since the early 1990s there has been a growing use of GIS software in research and development (Hartkamp, 1999). New issues have arisen with the growing use of GIS in research and development. One of these issues is the addition of time as a fourth dimension in GIS applications (Hartkamp, 1999). One of the ways the time issue has been addressed was by linking process based models to GIS applications (Hartkamp, 1999). Many of these models were created long before GIS applications ran on desktop computers. Consequently, they were created with simple input and output handling processes. As GIS software has evolved, application coding tools have become available allowing people to develop graphical user interfaces (GUIs). The GUIs are used to facilitate user interaction between creating input, executing models, and rendering output. However, understanding of the linkage between the model interface and the GIS interface that automates the input creation processes is still and
important issue. GIS applications have become powerful enough tools that they can be used to automate the model parameterization processes.

Manual parameterization of the process models can be an extremely time consuming and error prone process. GIS software allows us to automate most, if not all, of this process. However, often we see the demand for connecting more than one model to a GIS application and the models have different input and output format requirements. This is an example of where design patterns could play an important role in developing applications that link GIS and process models. By using a design pattern based approach for coding an application the linkage between the GIS software and the process model would be flexible enough to accommodate the addition of many process models without causing a domino effect in code changes throughout existing GIS software applications.

Consider the Strategy design pattern. Gamma (1994) defines the Strategy design pattern as family of algorithms, each one encapsulated and interchangeable. The Strategy pattern lets the algorithm very independently from clients that use the algorithm (Gamma et. al., 1994). Figure 31 is an example of the Strategy design pattern applied to the AGWA application.
The diagram shows the representation of two difference model input readers, KINEROS and SWAT. Each model has a unique input data format. The algorithm used to handle a model’s format is coded in either the KinerosInputReader class or the SwatInputReader class and other than implementing the input reader interface (IntfInputReader) the only code either class contains is the algorithm written specifically to handle the model’s input data format. The IntfInputReader interface acts as a bridge between a model’s parameterizer object and the parameterizer algorithm. The parameterizer object realizes its parameterizer process by implementing a “has-a” relationship with the input reader interface. With this pattern, a new model can be added to the system and the parameterization process can be incorporated by writing code specific to the new model without affecting code to any of the other models.
Another example where design patterns are critical in the development of an integrated GIS application is in presenting or managing model output. Similar to running the hydrologic model, the variation in input formats, model output formats also vary significantly. Employing the Strategy pattern, the programming would have a class interface with generic method calls for reading, writing, displaying and saving model output. The programmer would then write specific implementations that extend the class interface but implement the specific methods (i.e., read, write, display) unique to a particular output format.

In Figure 32 we see the Strategy design pattern used again to accommodate the different output formats used in each model.
The application code is written so that the KinerosOutputHandlers implement the display, read, and write methods, the IntfOutputProcessor acts as the bridge between the handlers and each model’s output processor.

Another design pattern that was important in the AGWA project was the Decorator pattern. The Decorator pattern assigns additional responsibilities to an object dynamically. The Decorators provide a flexible alternative to sub-classing an object in order to extend the object’s functionality (Freeman et. al., 2004). Decorators also provide a flexible alternative to sub-classing in order to extend functionality (Jia, 2002). In particular an area of applicability for the Decorator is when extension by sub-classing is impractical due to a significant number of different extensions, which would produce an explosion of subclasses to support every possible combination (Jia, 2002).
Within AGWA, the Decorator pattern became important when designing a strategy for handling the development of management plans by application users. Management plans have one or more outlet (delineation points), zero or more delineations, zero or more user-defined points, zero or more user-defined lines, and zero or more user-defined polygons. By applying the Decorator pattern to the management plan each plan can be constructed to accommodate the components described in simple or complex ways.

**Figure 33 Decorator Pattern Applied to Management Plans**

![UML Diagram](image)

Figure 33 is the UML model of the Decorator pattern applied to management plans. Management plans are defined by an abstract interface called the ManagementPlan. The ManagementPlanDecorator extends the abstract ManagementPlan class. Each concrete class (i.e., DelineationPlan) extends the decorator class (ManagementPlanDecorator). By extending the decorator each concrete class can maintain a “type-of” relationship with the ManagementPlan class, but also permits each concrete class to be implemented in ways
that make it unique or able to capture the attributes of a particular plan type while still making the object accessible without having to write specialty code.

**AGWA Ontology**

The combination of software design patterns and UML models defined for the DotAGWA application provide the foundation for developing the lingua franca for DotAGWA. Deliiska (2007) has four recommended steps for developing a geoinformatics ontology.

1. Common structure of domain ontology
2. Taxonomy of geoinformatics
3. Development of thesaurus and domain ontology of geoinformatics
4. Principles of ontology transformation for producing an application ontology

**Common Structure of Domain Ontology**

In terms of DotAGWA we see the common structure defined by the previously mentioned software design patterns and UML models. Graphically speaking, UML diagrams provide a common method for representing domain objects where each object is represented by a class diagram with methods (functions) and attributes (data).

Employing UML and software design patterns reveals the commonality between hydrologic models. Even though each hydrologic model may define physical space and interactions differently there are common functions required by each model that can be constructed in application (code) templates helping to re-enforce consistency and accuracy within the system (Freeman et al., 2004). The same applies to examining the
output produced by hydrologic model execution. The output data structure of each hydrologic model’s output files can vary significantly, but by applying a design patterns approach when examining the output structures common features are revealed. A simple example of the common features in hydrologic output is the fact that all hydrologic models currently employed in DotAGWA require file reading and writing to access the output data. Template functions can be created in interface classes that each model output handler can inherit. Using this technique helps maintain the flexibility of the system by facilitating the addition of future hydrologic models.

**Taxonomy and Thesaurus of DotAGWA**

For practicality’s sake we examine the taxonomy of DotAGWA and limit the definition of taxonomy to “classification; especially: orderly classification of plants and animals according to their presumed natural relationships” (Webster, 2008). The top level categories for the AGWA application are:

1. **Server Applications** – Supporting applications that provide data or access to other applications to the user or sub-components of the system. DotAGWA relies on two main server technologies. A web application server, Microsoft’s Internet Information Server, and ESRI’s ArcGIS Server. These two server components provide the bridge between the user interface and the backend supporting geodatabases, geoprocessing, and data manipulation.

2. **Client Applications** – The interface users access to interact with the system. In the case of DotAGWA this is a web client like Internet Explorer™. However,
AGWA, the desktop version of DotAGWA is dependent on the ESRI’s ArcGIS Desktop application.

3. **Process Models** – the process models integrated into the application. DotAGWA currently uses KINEROS and SWAT, but can integrate other process models. By incorporating a standardized framework for developing the application DotAGWA is flexible enough to be connected to many different models. By employing the Strategy design pattern the application delegates model parameterization and execution to the Strategy patterns bridging layer, which in tern lets the concrete class implementation that is unique to a model control how the model is parameterized and executed.

4. **Processing Functions** – This includes hydrologic geo-processing as well as data processing and formatting.
   
a. **Geo-Processing**
   
i. Watershed delineation – delineation of a watershed based on one or more user-defined outlet points
   
ii. Flow accumulation – a pre-processed geo-data set that determines the flow accumulation pattern using a digital elevation model
   
iii. Flow direction – a pre-processed geo-data set that represents the flow direction pattern using a digital elevation model
iv. Stream network – a pre-processed geo-data set representing the streams system

v. Watershed discretization – a delineated watershed that has been modeled to represent how a basin is viewed by the user’s selected watershed model.

vi. Model parameterization – A geo-data set formatted to store input data that is used by a user selected hydrologic model.

vii. Model execution – the component in DotAGWA that controls process or hydrologic model runtime.

b. **Data Processing**

i. Output reader – includes output reading processes for output produced by the hydrologic models

ii. Output formatter – processes used to handle hydrologic model output data. Formats these data sets to user or databases required formats.

5. **Input** – all input requires by the process models, server application, and client application. These supporting input data sets are initially established at application design time. However, application development processes can be quite dynamic. Consequently, as the application is constructed new data sources may be identified and will have to be analyzed for fit to the system and documented.
6. **Output** – Output typically generated by the process models, but also data created during intermediary processing steps. As with the input category, output formats should be accounted for at design time and new output formats may be required after the application development phase starts or even after the application has been developed. A flexible system accommodates the addition of new output formats.

**Principles of Ontology Transformation**

Ontology transformation is the slight change to the semantics of an ontology to make it usable for purposes other than those it was originally intended (Ding et al. 2001). Using this definition and considering the design framework (UML and design patterns) it is possible to extend DotAGWA to other data sets which would have to fit the input data requirements for the application. Geospatial data associated with the project is also available to the public through web services or applications able to transmit and receive spatial data delivered by various Internet communication protocols. This scenario does not necessarily mean DotAGWA is being used for purposes other than it was intended. However, it does provide a foundation for extending the framework. The most significant area where DotAGWA can be extended is by having the source code and design documentation be kept in the Open Source Software (OS) domain. Open Source software can facilitate the deployment of geospatial data and geo-processing functionality especially in a web environment like that of DotAGWA (Andersen et. al., 2004). By keeping the software in the OSS domain anyone can download the source code make
alterations and use the application as needed. The alterations can be shared with other users therefore extending the application or its components even further.

Finally, by employing the Model-View-Controller (MVC) design pattern the different components of the application can be re-used or re-combined with each other or other systems in order to satisfy the need of other applications. The MVC pattern consists of three kinds of objects. The Model is the application object, the View is its screen presentation, and the Controller defines the way the user interface reacts to user input. Before MVC, user interface designs tended to lump these objects together. MVC decouples them to increase flexibility and reuse (Gamma et al., 1994).

**NHDPlus Framework**

It is worth mentioning NHDPlus dataset/framework with regards to the proposed framework in this paper. The NHDPlus data set was developed from the United States Geological Society’s National Hydrography Data Set (NHD). NHD is collection of digital spatial data containing surface water information which includes lakes, ponds, streams, rivers, spring, and wells (USGS, 2008). NHDPlus data is an integrated set of application-ready geospatial data sets that incorporate NHD data, National Elevation Data (NED), and the National Watershed Boundary Dataset (WBD). NHDPlus also includes stream network features as well as value added attributes from other data sets including land cover. NHDPlus is useful in that it is a standardized data set that is gaining popularity in many different research projects. Also, much of the data was processed or constructed using the ESRI’s ArcGIS Hydro Data Model (Maidment, 2002).
By fitting the Hydro Data Model NHDPlus can be easily integrated into ESRI’s ArcMap application and easily analyzed using ESRI’s ArcHydro tools (Maidment, 2002).

NHDPlus did not become available until around 2005 and has useful features that could be incorporated into the AGWA and DotAGWA applications. However, because of the timing of the NHDPlus data availability it was not incorporated into the application. However, NHDPlus has generated new ideas for the DotAGWA project. The NHDPlus data are based on the ArcHydro model, which essentially builds a relational database for a watershed. This relational data structure could be constructed for the data sets used in AGWA/DotAGWA and they would be likely to increase geo-processing speeds. Geo-processing speeds would increase because the processes would switch from raster data sets as input data to vector data sets as input. A second useful idea that could be used in future versions of AGWA/DotAGWA would be pre-processing the river network using ESRI’s ArcHydro tools. The advantage with this change would mean minimizing some of the raster-based geo-processing functions. Finally, one possible drawback to using NHDPlus data itself lies in the NHDPlus developed stream network. A stream reach in a sub-basin should be one contiguous line without any line segments. NHDPlus data may have streams at the sub-basin level that have line segments. This situation would cause problems with relation to parameterizing hydrologic models. The existing AGWA/DotAGWA application code expects contiguous line segments in the stream network.
SUMMARY AND CONCLUSIONS

In particular the 1990s saw a decrease in cost for computer processing power, which was related to the increase in the use of GIS applications for research and development (Harthkamp et al., 1999). These two things together lead to a new generation of research problems more pertinent to researchers than to programmers and GIS developers in an effort to expand on this issue this paper has discussed using UML, Software Design Patterns to address the issue of developing an application framework that unites geospatial information and hydrologic models. The need for linking these two technologies has seen an increase in demand due to factors related to limited water resources, accelerated growth of cities and towns, greater demands on limited resources, but also from increased in computing power, computer processing speeds, data storage capacity increases and advancements in geo-processing functions.

By developing a few simple UML models, describing software design patterns used in the DotAGWA application, addressing issues and trends in linking GIS applications and process models similar to those associated with DotAGWA, and by developing a simple ontology for the DotAGWA system this paper presents these items as a possible solution to developing the language and communication the stakeholders involved in this kind of project could use to help guarantee the success, acceptance of an application. The techniques here also serve as an example of a development methodology that serves as a way to increase the lifetime of an integrated process model/GIS application. Finally, and
most importantly the processes described in this paper and used during the development of the DotAGWA application provide a tool that can enhance the quality of a research project by providing a common and standardized language that serves as the foundation for developing similar research projects. The shared language, through UML and Software Design Patterns provide the tools to eliminate ambiguities in communication without confining the communication and risking the loss of information.
REFERENCES


