EnviroInfo 2007 (Warschau)
Environmental Informatics and Systems Research
 ISBN: 978-3-8322-6397-3

Extending GIS-Functionality for the Analysis of 3D Data in Landscape Ecology – A Comparison of Different Approaches

Nguyen Xuan Thinh[1], Le Hai Dang[2], Sebastian Hoechstetter[1], Ulrich Walz[1]

Abstract

Powerful tools for the analysis of spatial data and the characterisation of landscape patterns are required for a sound understanding of ecological processes. This particularly holds true for 3D-data, as topography and surface of vegetation and human artefacts plays an important role in the geomorphological, hydrological, and biological processes present in the landscape. This paper discusses the necessity to integrate mathematical methods and algorithms into geographic information systems (GIS), especially ArcGIS from ESRI®, in order to develop 3D-indices in landscape ecology. Moreover, different ways to extend the analytical capacity of standard GIS are described and compared. A demonstration of programming in .Net will be given. Finally, the third section reports about some examples for applications which have been conducted in a study financed by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG).

1. Introduction – necessity to incorporate complex calculation routines into GIS-analyses

The rapid development of GIS has enabled the combination or integration of different methods and algorithms (models). The integration of methods and models into GIS may help to create powerful tools in environmental and geographical sciences. The GIS itself becomes an integral part of those models by providing both the mapped variables and the processing environment.

In particular, the analysis of 3D-data in landscape ecology requires the power of GIS to perform input, output, and basic housekeeping functions, such as preliminary resampling or transformation of data as well as modern mathematical algorithms. In such combinations, GIS usually fulfils tasks that can be characterised as preprocessing or postprocessing. Preprocessing includes coordinate transformation and projection change, resampling or conversion between data models and structures, windowing and clipping to fit study areas, and analysis and modelling of uncertainty. Postprocessing includes cartographic and visual display, simple spatial analysis of results, verification, and analysis of outliers and residuals.

Advances in mathematics, e. g. wavelet analysis, signal processing, spectral analysis, fractals, lacunarity, and their applications in landscape ecology, however, oftentimes cannot be implemented using standard desktop-GIS-functionality. Also special software for relief analysis (e.g. SAGA) doesn't include all of the functions needed. Therefore, linkage and integration of modern mathematical methods and algorithms into standard GIS like ArcGIS from ESRI® as well as providing user-friendly analysis tools is necessary and requires a synergy of mathematics, (geo)informatics and landscape ecology. The next section will discuss different approaches to extend standard GIS-functionality.

[1] Leibniz Institute of Ecological and Regional Development, Weberplatz 1, 01217 Dresden
[2] Technische Universitaet Dresden, Department of Computer Science
e-mail: ng.thinh@ioer.de, lhdang@hotmail.de, s.hoechstetter@ioer.de, u.walz@ioer.de,

2. Different approaches to extend standard GIS-functionality

2.1 Survey of different ways

If we think of extending standard GIS-functionality we can at first imagine the following ways to do so:

1. Use ModelBuilder embedded in ArcGIS,
2. Program in script language like Arc Macro Language (AML), Python, VBScript or JScript,
3. Write VBA code,
4. Develop COM (Component Object Model) components by using an external programming platform, e. g. .Net (C# or VB.Net), C++ or Java Enterprise Edition (Java EE) and integrate them into ArcGIS,
5. Coupling GIS and other software programs, e. g. linkage ArcGIS to MATLAB
6. Development of own specific GIS

(1) The ArcGIS ModelBuilder provides a graphical modeling framework for designing and implementing geoprocessing models which can include tools, scripts, and data. Models are data flow diagrams that link together a set of tools and data to create advanced procedures and work flows. ModelBuilder is a productive mechanism to share methods and procedures with others, both within or outside a project team. But ModelBuilder is not suitable for the development of complex applications.

(2) Each script language with COM-interpreter such as AML, Python (syntax like C and Pascal), VBScript (based on Visual Basic, often used as a web programming language for inter.Net explorer), and JScript (syntax like C and Java) can be used to communicate with the Arc Environment and to build programs of frequently user ArcInfo commands, or menu-driven applications (see for example Thinh & Hedel, 2004). In general, an interpreted language is considerably slower than a compiler language like C, C++ or FORTRAN. The main disadvantage of such languages is that they are not powerful enough for programming complex problems.

(3) VBA (Visual Basic for Applications) is embedded in ArcGIS, and enables the development of specific applications. VBA is a programming language, but it also serves as a macro language. It provides low level data access and is powerful for programming of less complex extensions. A benefit of programming in VBA lies in the master property of the language. One can adapt available standard macros for other applications. However, VBA has disadvantages in speed and productivity (in terms of writing code for medium-size and large-size applications) compared to the process of creating extensions with external programming platforms and lacks an powerful integrated development environment IDE (such as Eclipse for Java or MS Visual Studio for .Net). In addition VBA as a technology is over 10 years old, and it is being slowly replaced by a new programming platform, the Microsoft, .Net Framework.

(4) .Net is a new kind of software platform and programming tools to make building constellations of multiple applications, devices and services faster and easier. One advantage of .Net is a high level of interoperability. This platform provides means to access functionality which is implemented in programs that execute outside the .Net environment and allows objects from different languages to operate together (Bradley & Millspaugh, 2006, p. 5). A key part of .Net is the Common Type System (CTS) with the functions:

- to establish a framework that helps enable cross-language integration, type safety, and high performance code execution,
- to provide an object-oriented model that supports the complete implementation of many programming languages, and
- to define rules that languages must follow, which helps ensure that objects written in different languages can interact with each other.

Currently there are over 40 so-called .Net languages which are used to produce programs that execute within the .Net Framework. But only a small number of them are widely used and supported by Microsoft. Microsoft provides several such languages, e. g. C#, VB.Net, and C++/CLI (Common Language Infrastructure). C# (C Sharp), Microsoft's flagship .Net Framework language, is an object-oriented programming language developed as part of the .Net initiative, and later approved as a standard by ISO and the European Manufactures Association (ECMA). The procedural and object-oriented syntax of C# is based on C++ and includes aspects of several other programming languages (e. g. Delphi and Java) with a particular emphasis on simplification. Therefore C# and Java have many similarities, even if they are strong competitors. Both are based on a virtual machine model which hides the details of the computer hardware. C# as well as Java use their own intermediate byte-code. On .Net, the byte-code is always just-in-time (JIT)-compiled, also known as dynamic translation. This is a technique for improving the runtime performance of a computer program. It converts, at runtime, code from one format into another, for example byte-code into native machine code. With Java, the byte-code can be either interpreted or JIT-compiled. Some differences between .Net and Java can be mentioned. .Net was built to support multiple programming languages while targeting Microsoft Windows, whereas the Java platform was built to support only the Java Language on many operating system platforms. .Net in its complete form is currently only available on Windows platform, while Java is fully available on many platforms.

The choice of an appropriate programming platform (VBA, .Net, C++ or Java) depends fundamentally on following criteria: programming knowledge, programming effort, performance, documentation/development support and licence costs. The programming effort depends especially on the existing programming knowledge, because a period of vocational adjustment for the technologies COM, ArcObjects, programming languages is required. The platforms .Net and Java have been developed for a fast development of code. In comparison with older programming languages like C++ or Visual Basic, the handling of these modern programming environments is simpler and one can create complex applications whose code is more structured and easy to extend. This is achieved by a stricter object oriented programming concept (all classes inherit from the upper class object, no multiple inheritance, no macros etc.), thus for example one can create clearer and well arranged code. In addition, using these platforms a higher productivity can be reached, because the programmer has to attend less to the memory management which has been conducted by the garbage collection automatically (Watkins et al. 2003, p. 145).

Hunt (2002) stressed that C# is not Java and is not C++ and that C# introduces a number of new concepts (for a comparison of C# to Java and C++ see e. g. Hunt (2002), p. 50-51; for specific hints to compare of C++ to Java see online resources by Josutties (2003). In general the developer for the period of vocational adjustment in C# needs less time than in the case of C++ because C++ is a more mighty language and can overburden a beginner with its extensive possibilities. Particularly for debugging and memory management one must have a sound knowledge about the language. With the integrated Application Programming Interface (API), .Net offers comfortable ways for creating of own windows and dialogs under windows. Using C++, the programming of user interfaces is more difficult, as one has to access to the WinAPI or to Microsoft Foundation Class (MFC) Libraries and must have deeper knowledge for Windows programming. However a little lower performance must be accepted when using of .Net and Java in comparison with C++ and Visual Basic since .Net and Java code unlike C++ and Visual Basic code is not converted into machine code directly, but into a intermediate code in the Common Intermediate Language (CIL) or in the bytecode (Java). This intermediate code needs a runtime environment for the execution (.Net Common Language Runtime, Java Runtime Environment) which runs in parallel and converts the code into machine code in the background. The advantage of this is the possibility to write safe code and to be platform independent (Kühnel, 2006).

The two approaches (5. and 6.) to extend the standard GIS should be not detailed described in this paper. The best-known forms of coupling of GIS to other software programs are loose coupling and tight coupling (Steyaert & Goodchild, 1994). Loose coupling combines the capabilities of separate modules for GIS functions and other software programs by transferring simple files. Tight coupling allows GIS and

the software program to run simultaneously and to share a common database. The software SAMT (Wieland et al. 2006) is an example for the development of own specific GIS.

From all points of view, the .Net platform offers the simplest and most effective way for the development of ArcGIS extensions (C# is easier to learn than C++ and more modern than Visual Basic and supported by ESRI and Microsoft). Therefore in the section 2.2 we describe a demonstration of programming in C# under the .Net platform in order to develop an ArcGIS extension for the analysis of 3D-data.

2.2 Programming in .Net – A demonstration

Generally one has to work with ArcObjects when trying to develop an ArcGIS extension. As all ArcDesktop applications are built on ArcObjects, the programming-process to extend of ArcDesktop functions can be conducted according to a common procedure.

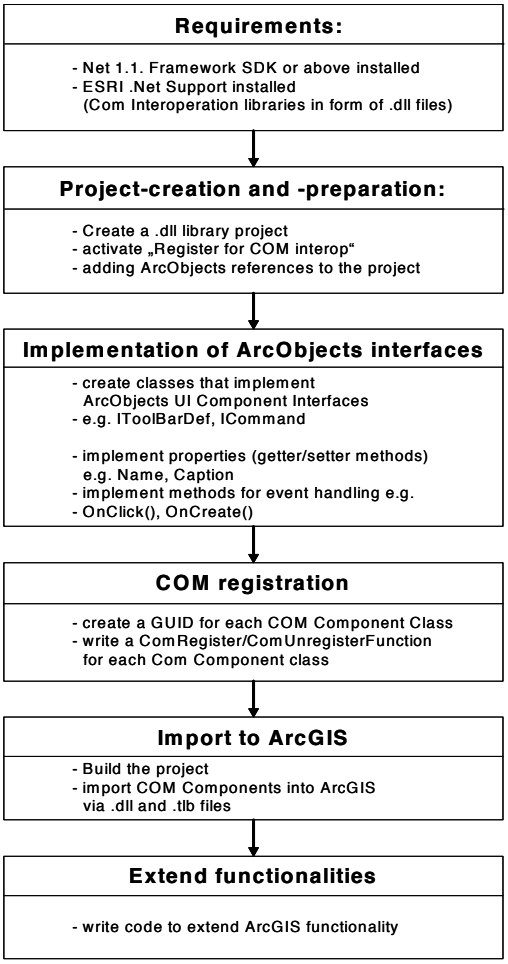

Figure 1: General procedure for the development of an ArcGIS extension

Figure 1 presents the common steps of the programming of an extension by using C# under the .Net situation.

As a demonstration example we will extend ArcMap with a custom toolbar, from where we can start the extension-code (extension-code is our code for adding functionality to ArcMap). Thus the Toolbar acts as an entry-point for the extension-code. The Toolbar will be declared as a COM Object so that it can be plugged into ArcMap.

(1) The requirements for the development under .Net are:
- to have at least .Net Framework 1.1 and Software development kit (SDK),
- the COM-Interoperation libraries for .Net are installed (provided by ESRI), and
- an Interactive Development Environment (IDE) e.g. Visual Studio 2005 is available.

(2) Project-creation and -preparation
First we have to create a new project. Since an extension is a COM component we have to choose a .DLL project, which generates a .DLL and a type library file as results. The .DLL file will contain the COM components that we create. In the project properties (in the Project tab) we must activate "Register for COM interop", to allow our COM components to be registered in the Windows registry, thus enabling ArcMap to use the COM component. The last preparation step is usually adding basic references (the basic libraries of ArcObjects) to the project .that allow the use of ArcObjects functionality. The references we basically need here are:
- ESRI.ArcGIS.ArcMapUI
- ESRI.ArcGIS.SystemUI
- ESRI.ArcGIS.Framework

(3) Implementation of ArcObjects interfaces
After all preparations are done, we will create classes that contain the code for plugging in a toolbar and a command to ArcMap. Because a toolbar is just a container for storing a user interface component, we need to create a command which actually is a user interface component (button) too that can start an action on a clicking-event. The command should actually start the extension-code and be added to the toolbar.

Therefore two classes must be implemented. Each class implements an interface representing its type. The toolbar class implements the interface IToolBarDef from ESRI.ArcGIS.SystemUI and the command class implements ICommand from ESRI.ArcGIS.SystemUI. The interfaces define the way how the classes must look like, to be compatible within ArcMap. For example the prop-erties "Name", "Caption" of the IToolbar interface should be implemented so that they return the name or the title of the toolbar.

(4) Registration of COM components
In order to register a class as a COM component, the class must have a "Globally Unique Identifier" (GUID) and a register and unregister function. In .Net the GUID number can be generated by the provided tool GUIDGEN.EXE (see Microsoft Developer .Network). The COM-register and unregister functions are declared in .Net by using the [ComRegisterFunction()] attribute:

```
//C# code
[ComRegisterFunction()]
static void NameofRegisterFuntion(String key)
{code for registration...}
[ComUnregisterFunction()]
static void UnregisterFunction(String key)
{code for unregistration...}
```

(5) Import to ArcGIS

After this implementation we can build the project. The compiler will create a .dll (dynamic linked library) and a .tlb (type library) file that contains the 2 COM components and the extension-code. By activating the "Register for COM interop" checkbox in the preparation step, the compiler automatically registered the toolbar and command classes to the Windows registry by using the ComRegisterFunction. The code implementation process is therefore complete and the extension has to be imported into ArcMap. To do so we just have to start ArcMap and choose *Tools* ➔ *Customize* ➔ *Add from file* and choose the .tlb file. The Toolbar and the Command will be then imported in ArcMap. They now are ArcMap UI components and thus can be drag and dropped to other UI Components just like we used to do it with other components. The Extension will also be started on every time when ArcMap starts up.

(6) Extend functionalities

As for the ICommand Interface the method OnClick() defines the behaviour of the Command when clicking on it. This method can be used to be the entry-point of the extension code, for example by creating a new Window that represents the extension itself. From this window every possible code can be called, for example accessing a Raster-dataset by calling the appropriate COM function and run a simple moving-window algorithm, the result of the calculation can then be visualized by some graphic libraries (e.g. OpenGL or .Net "Graphics" library). Another example could be that the window provides editing functionality on geodatabase data in form of tables. Each time the data has been edited, the extension code could force ArcMap to instantaneously draw the new data on the display. The code that can be called by the command is therefore not restricted (e.g. doing parallel calculations, .Networking etc.), meaning that we can also use other 3rd party libraries to build up our extension.

3. Examples for applications

In a study financed by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG), the attempt was made to achieve an integration of the third dimension into analyses of landscape structure; in previous approaches, ecologically meaningful 3D-structures like terrain shape or elevation are not taken into account and valuable information is lost (Hoechstetter et al. 2006; for a visualisation of this effect, see Figure 2).

For this purpose, extensive analyses using digital elevation models (DEMs) of different horizontal reso-lutions (down to 1x1 m for DEMs from Airborne Laser Scanning) were carried out. Thus, very large raster datasets had to be processed, using rather advanced calculation algorithms demanding high computational efforts. Some spotlights from this study are supposed to illustrate the use of the different customisation-approaches mentioned in the previous section.

For example, simple moving-window-algorithms to calculate "landform grids" from DEMs were computed. Using the formulae of McNab (1992) or Blaszczynski (1997), these algorithms may reflect the "concavity/convexity" of the surroundings of a focal raster cell and thus provide a simple measure of curvature. Standard ArcGIS-functionality does not provide the flexibility needed to perform moving-window-algorithms based on formulae which are not included in the menu of "FocalStatistics"-toolbox of ArcGIS Desktop-environment. But those routines could be implemented using the ArcObjects-library and the VBA-environment. Advantages of this way of implementation are the availability of basic VBA sample code for filtering algorithms that may easily be altered to suite the individual needs as well as the flexibil-ity to perform computations based on various formulae and neighbourhoods.

For other analysis techniques, like approaches from fractal mathematics or signal processing, the Matlab environment (MathWorks 2005) with its modular setup proved to be a suitable supplement to established GIS software. For example, wavelet analysis may be a suitable technique to test digital elevation

models for certain features like recurring patterns or texture (see Dale & Mah 1998; Saunders et al. 2005). The straightforward import of different raster data types and the large library of functions and scripts enable the user to perform such analyses. Also lacunarity analysis, a principle having its seeds in fractal mathematics and used within the project to examine the variability and heterogeneity of landscape sections (see Hoechstetter & Walz 2006), was carried out mainly using the scripting language coming along with Matlab. As the major disadvantage of this software package, the frequently long computing times for large datasets can be mentioned.

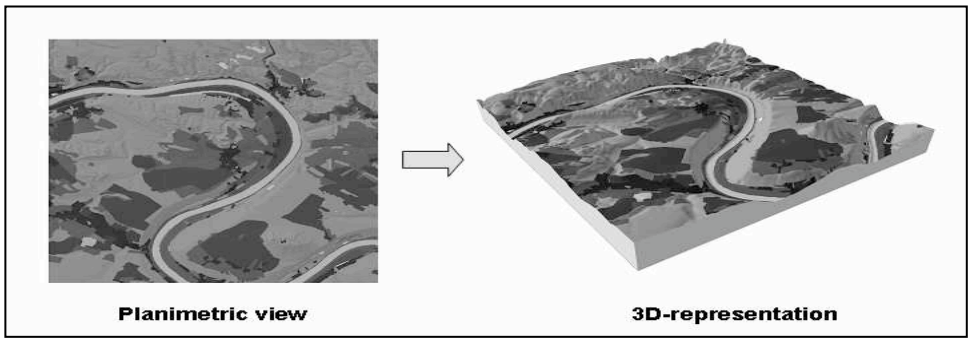

Figure 2: Composition and configuration of a landscape mosaic in planimetric and three-dimensional view: analysis of area and perimeter of single landscape elements and the distances between them leads to differing results for the two representations. (The section displays an area in the valley of the river Elbe in Saxony/Germany).

Finally, a computational framework was needed for the implementation of a more comprehensive concept of landscape ecological research: the analysis and description of landscape structure. This is one important component of landscape ecology. According to this concept, landscapes can be characterized by a typical composition and configuration of single landscape elements ("patches") forming a "landscape mosaic". So far, widely used programmes like Fragstats (McGarigal & Marks, 1995) are used to compute a large variety of landscape metrics, which can provide numerical information about the structuring of landscape mosaics. Still, those approaches act on the assumption of the land surface as a planimetric ("2D") layer. Relief and landform are not taken into account. Thus, part of the study at hand was to incorporate the relief into landscape structure analyses and to correct such landscape metrics for effects of elevation and landform. This required are more intricate implementation. A self-designed program ("extension") for ArcGIS Desktop turned out to be the most effective technical approach to do so. This extension was pro-grammed using the .Net Framework and the programming language C# as outlined in the previous section 2.2. This allowed for a complete integration of new and rather advanced functionality into the GIS-environment and user-friendly handling and export of analysis outputs. The linkage of the assemblies and functionalities provided by the .Net Framework and the ArcObjects library proved to be ideal for this task, while the implementation of course was rather time-consuming and elaborate.

Those examples from this ongoing research project are supposed to highlight the potentials and handicaps of some different ways to perform analyses of spatial data for problems in landscape ecology that cannot be carried out using standard GIS functionality alone.

Acknowledgements

The authors would like to thank the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) for the funding of this project.

Bibliography

Blaszczynski, J.S. (1997): Landform characterization with geographic information systems. Photogrammetric Engineering & Remote Sensing, 63(2): 183-191.

Bradley, J.C., Millspaugh, A.C. (2006): Programming in Visual Basic .Net. McGraw-Hill, Boston.

Dale, M.R.T., Mah, M. (1998): The use of wavelets for spatial pattern analysis in ecology. Journal of Vegetation Scence, 9: 805-814.

Hoechstetter, S., Thinh, N.X., and Walz, U. (2006): 3D-Indices for the Analysis of Spatial Patterns of Landscape Structure. Reviewed paper in: Kremers, H., Tikunov, V. (Eds.): InterCarto-InterGIS 12. International Conference on Geoinformation for Sustainable Development, Berlin, Germany, August 28-30, 2006. Proceedings, Berlin: Deutsche Gesellschaft für Kartographie, 108-118.

Hoechstetter, S., Walz (2006): Werkzeuge und Methoden zur Analyse von dreidimensionalen Landschaftsstrukturen. In: Wittmann, J. & M. Müller (eds.): Simulation in Umwelt- und Geowissenschaften - Workshop Leipzig, Shaker Verlag, Aachen, 235-244.

Hunt, J. (2002): Guide to C# and Object orientation. Springer, London.

Josuttis, N. (2003): Object oriented programming in C++, Chichester, Wiley.

Kühnel, A. (2006): Visual C#2005. Galileo Computing, Bonn.

MathWorks (2005): Matlab. Natick, Massachusetts.

McGarigal, K., Marks, B.J. (1995): FRAGSTATS: spatial pattern analysis program for quantifying landscape structure. Portland.

McNab, W.H. (1992): A topographic index to quantify the effect of mesoscale landform on site productivity. Canadian Journal of Forest Research, 23: 1100-1107.

Saunders, S.C., Chen, J., Drummer, T.D., Gustafson, E.J., Brosofske, K.D. (2005): Identifying scales of pattern in ecological data: a comparison of lacunarity, spectral and wavelet analyses. Ecological Complexity, 2: 87-105.

Steyaert, L.T., Goodchild, M.F (1994): Integrating geographic information systems and environmental simulation models: A status review. In: Michener, W.; Brunt, J; Stafford, S. (Eds.): Environmental information management and analysis: ecosystem to global scale. Taylor London, 333-355.

Thinh, N.X., Hedel, R. (2004): A Fuzzy Compromise Programming Environment for the Ecological Evaluation of Land Use Options. Proceedings of the 18th International Conference Informatics for Environmental Protection, CERN Geneva, Vol. I, Editions du Tricorne, 614-623.

Watkin, D., Hammond, M.; Abrams, B. (2003): Programming in the .Net environment. Addison Wesley, Boston.

Wieland, R., Voß, M., Holtmann, X., Mirschel, W., Ajibefun, I. A. (2006): Spatial Analysis and Modeling Tool (SAMT): 1. Structure and possibilities - Ecological Informatics, 1(1): 67-76.