Zaragozi, B. M.; Belda, A.; Linares Pellicer, JJ.; Martínez-Pérez, J. E.; Navarro, J. T.; Esparza Peidro, J. (2012). A free and open source programming library for landscape metrics calculations. Environmental Modelling and Software. 31:131-140. doi:10.1016/j.envsoft.2011.10.009.

A free and open source programming library for landscape metrics calculations: land-metrics DIY.

Zaragozí, B.[1]; Belda, A.[2]; Linares, J.[3]; Martínez-Pérez, J.E.[2]; Navarro, J.T. [1]; y Esparza, J. [3]

1 Institute of Geography, University of Alicante
2 Natural Resources Mapping Unit, University of Alicante
3 Department of Informatic Systems and Computation, Polytechnic University of Valencia

Abstract

Landscape metrics are used in a wide range of environmental studies such as land use change and land degradation studies, soil erosion and runoff predictions, management of hunting communities, and strategic planning for environmental management, to name but a few. Due to their utility for a variety of applications, there are many indices and software packages that have been designed to provide calculations and analysis of landscape structure patterns in categorical maps. With the purpose of making a profound comparison between the most used tools (Fragstats, V-Late, PA4…), we examined their advantages and disadvantages in order to create a list of common features that need to be incorporated into this type of software. We believe that an API without limitations on data input is necessary, capable of calculating vector or raster metrics and very extensible. This API should make it possible not only to build third party applications in easily, but would also make it possible to add new metrics and research into new paradigms related to traditional landscape metrics. We have started to develop a proposal based on open standards, which is FOSS. We have called this API Land-metrics DIY (Do It Yourself). It can calculate almost 40 landscape metrics from geometry provided by an ESRI Shapefile, but we are working to complete its contents as we explain in this article.

Keywords: FOSS, Fragstats, GIS, C#, Land-metrics and Landscape Ecology.

1. Introduction

1.1. What are landscape metrics?

It is necessary to start by defining the subject under study in this paper. In the bibliography, we can find two similar terms. The term "landscape indices" is more frequently used in a broader sense. On the other hand, the term "landscape metrics" appears more frequently, but there are no definite rules or traditions as to when one or the other term is used (Uuemaa et al., 2009). In this paper we have decided to use the latter.

Common usage of the term "landscape metrics" refers exclusively to indices developed for categorical map patterns. Landscape metrics are algorithms that quantify specific

spatial characteristics of patches, classes of patches, or entire landscape mosaics. Thus, landscape metrics indicate spatial patterns that reflect differences in dominant factors that configure the landscape (Matthew et al., 2009). A plethora of metrics has been developed to quantify categorical map patterns. These metrics fall into two general categories: in the first place, those that quantify the composition of the map without reference to spatial attributes, and secondly, those that quantify the spatial configuration of the map, which require spatial information for their calculation (Gustafson, 1998; McGarigal and Marks, 2002).

Many landscape metrics applications come from the field of Landscape Ecology, which is the science of studying the relationship between spatial pattern and ecological processes on a multitude of landscape scales and organizational levels (Wu, 2006).

1.2 Landscape metrics applications

As has already been mentioned, Landscape Ecology provides many methods to study the composition and configuration of habitats as a compilation of discrete patches. These methods need to be adapted for wildlife species in different environments (Le Pichon et al., 2009).

Landscape Ecology provides an extensive set of indicators to evaluate several processes related to environmental issues. The application of landscape metrics has provided good results in the context of land degradation studies (Simoniello et al., 2006). The analysis of spatial heterogeneity in vegetation and soil properties can be used to improve predictions of runoff and erosion (Lesschen et al., 2008). Also, landscape metrics help to model watershed hydrological systems and to identify and analyze possible future impacts on land use pattern and hydrology (Lin et al., 2007). There are studies based on the relationship of landscape structure with the hunting community (Jimenez-Garcia et al., 2006), as well as specific studies on the wild boar (Calenge et al., 2004; Hebeisen et al., 2008; Kaden et al., 2005; Monzón & Bento, 2004; Tsachalidis and Hadjisterkotis, 2008), red-legged partridge (Nadal, 2001; Vargas et al., 2006), ducks (Guillemain et al., 2008), mouflon (Garel et al., 2005), wild rabbit (Schröpfer et al., 2000) and some predators (Rico and Torrente, 2000). Landscape metrics enable this information to be incorporated into a GIS, helping to select potential areas and take appropriate management measures (Coulson et al., 2001).

Maintaining and restoring landscape connectivity is an increasingly central concern in ecology and biodiversity conservation, and there is an increasing demand for user-driven tools for integrating connectivity into landscape planning (Saura and Torné, 2009). A new approach suggests alternatives to the traditional patch mosaic model that considers situations where spatial heterogeneity is continuous rather than discrete. Thus, habitat is viewed as a continuous gradient instead of discrete patches within a homogeneous matrix. Moreover, heterogeneity is viewed as a three-dimensional surface and can represent any ecological attribute of interest (Hoechestter and Walz, 2009; McGarigal et al., 2009). This new approach provides greater accuracy than the previous "2D" metrics.

1.3. Several considerations about calculating land-metrics

Landscape is not necessarily defined by its size; rather, it is defined by an interacting mosaic of patches relevant to the phenomenon under consideration at any scale. On the other hand, when studying wildlife, multiscale analysis can provide insight into the spatial scale at which species respond, a topic of intrinsic scientific interest with applied implications for researchers establishing protocols to assess and monitor wildlife populations (Brennan and Schnell, 2005).

The accuracy of landscape analysis depends on spatial and temporal scale and these are characterized by data format. Most landscape structure measurements can be calculated using either raster or vector data formats and processing methods (Wade et al., 2003). GIS-based measurements that combine native raster and native vector data are commonly used in environmental assessments. Evaluations often cover large areas, and metrics are usually calculated using raster methods. Raster processes are more commonly used because they can be significantly faster computationally than vectors, but error is introduced in converting vector data to raster. For assessments based on rankings or groups, results indicate that any of the methods are sufficient. If highly accurate individual observations are required, vector methods should be employed when possible. Assessment needs will determine which processing method is appropriate for a given metric. When the reporting unit is large relative to the pixel size, the method will have little or no impact on the assessment and the raster method is preferred for its greater efficiency. For many assessments, the faster raster or hybrid methods will provide adequate results, especially when buffer size is large (Wade et al., 2003).

1.4. Available software for calculating landscape metrics

Once the utility of calculating land-metrics in many landscape ecology studies and in other scientific fields has been accepted, it is necessary to obtain an appropriate tool adapted to our practical interests.

Several software packages provide methods for analyzing landscape patterns observed in raster grids and remote sensed images, because it has been found that the quality of classification of land-uses can be improved considerably by the use of structure analysis, which may utilize measures of the kind described below or a range of other forms of surface variability analysis.

There are many tools specifically developed for calculating landscape metrics, highlighting Fragstats, (McGarigal et al., 2002; McGarigal and Marks, 1995). Nevertheless, because of the necessity of handling spatial data (Longley et al., 2005; Steiniger and Weibel, 2009; Turner et al., 2001), many modules integrated in GIS desktop software exist, for example V-Late and PA4 in ArcGIS, Pattern and Texture modules in IDRISI, and at least two GRASS packages (r.le and r.li). The use of GIS provides all of the additional advantages of using this type of software, such as many available data formats, geoprocessing tools, data editing and many report possibilities. Despite the advantages of calculating land-metrics with a GIS, obviously, it is necessary to be familiar with the GIS being used. These desktop applications are for general purposes, which means that we have to adapt our workflow to the tool. Another problem is that all of these tools work in different ways, and do not calculate the same metrics.

Table 1: Comparison of the best known landscape metrics tools.

Program	FOSS	Platform	Programming language	Other requirements	Data format
Fragstats 3.3	No (yes on v.2)	Windows, Mac(v.2)	C++	Ext. to GIS	Raster
V-Late 1.1	No	Windows	VB 6	ArcGIS 9.x	Vector
Patch Analyst 4 beta	No	Windows	VB 6	ArcGIS 9.x	Both
r.le + r.li	Yes	Windows, Mac & Linux	C	Grass 6.4	Raster
Pattern & Texture	No	Windows	----------	Idrisi Taiga	Raster
IAN	Yes	Windows*	Ruby	----------	Raster

In Table 1, the main differences between the most popular land metrics tools can be seen. We do not intend to undertake a complete review of all the existing software, but it is necessary to know some details in order to evaluate various pros and cons, and to obtain a global picture of the state of the art in these techniques. Of course there are many specific tools that we are not going to consider because of their specificity.

In the first place, Fragstats is the most popular software for the calculation of landscape metrics. It was released in the public domain in 1995 (version 2), and was updated in 2002. This software is probably the best project of those considered in this comparison, it has very complete documentation on its webpage, and calculates more than 100 metrics and, of course, it is the most complete program commented on here. This software works by always using raster models and formats, and a good point is that it is freely available. Its latest version is not FOSS and only runs on Windows. Its workflow requires some pre-processing tasks to import images, rasterize vector files, create a class file and then it is necessary to configure some menus. These pre-processing tasks are more time consuming than in other computer applications.

As for implementations on the ESRI platform, there are two ArcMap extensions for calculating some landscape metrics, V-Late and PA4. The first, V-Late (Lang and Tiede, 2003) is an extension that calculates some vector formulas. It has many advantages over other software as it is implemented on a very well known desktop GIS which allows building on the available formulas using the Arcobjects® programming potential. In this case, the software is not FOSS, and needs at least an ArcGIS user licence to run. With regard to its workflow, V-Late reports are not customizable enough. The patch metrics are written into the related table of the vector file, and the rest of the results are output in several loosely structured text files that may collect redundant information. Moreover, it is necessary to build a new vector file to get the Core-metrics results, and this task does not work with large numbers of polygons.

On the other hand, PA4 (Rempel and Kaufman, 2008) is the most recently developed tool based on ArcGIS and like V-Late is an extension programmed in VB6. This language is unsupported by Microsoft since April 8, 2008 (see the Support Statement in the MSDN, Visual Basic 6.0 Resource Center; http://msdn.microsoft.com/en-us/default.aspx). This can cause some problems and incompatibilities when working with newer technologies. These programs will have to adapt a few years after birth, or die very young. In spite of this, PA4 calculates many landscape metrics using vector

and raster formulas. The raster formulas are calculated through a Fragstats interface. PA4 provides many extra tools approaching and giving direct access to many ArcGIS capabilities. Moreover, the Core-metrics calculations work better than V-Late with large numbers of polygons.

IDRISI is a well-known and affordable GIS, with considerable raster processing capabilities. Like ArcGIS, IDRISI makes it possible to develop new modules using several programming languages, but it is not open source software. It has two modules that can be used for calculating some land-metrics from rasters. Directly, without considerable programming effort, it calculates fewer metrics than the other software considered here (Cartwright, 1991).

Perhaps one of the most interesting GIS based applications is GRASS and its specific modules r.le (Baker and Cai, 1992; Baker, 2001), and r.li (Porta and Spano, 2008). Using these modules it is possible to calculate a large list of metrics, but always with raster data inputs. As in Fragstats, if data is in a vector format it must be rasterized first. As an OSGeo project, Grass is FOSS, overall programmed in ANSI-C or Python too, and it can run on Mac, Linux or Windows platforms. The raster format limitation mentioned for these modules can be overcome thanks to the GRASS vector capabilities, but at the expense of some programming effort (Wang, 2008).

Finally, there are two projects that have been developed by the Forest Landscape Ecology Lab at the University of Wisconsin-Madison: APACK and IAN. The latter has replaced APACK. IAN is a FOSS project developed with Ruby, which is an interpreted OOP language that is very easy to learn. Being OOP, it has many advantages over all the other programming languages referred to here before. IAN only works with raster formats, and reads a few GIS raster formats, although by programming, the available raster formats accepted could be increased. The main web page of the project (http://landscape.forest.wisc.edu/projects/IAN/) states that it runs on Windows, but Ruby has interpreters in many platforms and it could be possible to use IAN with other different OS. Perhaps the only drawback of this project is that it is completely independent from other FOSS GIS projects, which makes it more difficult to include contributions from external projects.

1.5. Considerations about existing tools. Why a new tool is needed.

After this general description of commonly used programs it is possible to sum up the most important goals that a useful program must accomplish:

(1) A specific tool is required, not a lot of generic ones. An application designed specifically for calculating landscape metrics will be more productive than using a general purpose GIS application or creating programs for each metric. This will make coping with several desktop GIS unnecessary, as none of them integrates all possible metrics, or integrates different formula implementations for the same metric. In this sense, Fragstats is the most mature tool, and may be an example of specificity (Wang, 2008).

(2) FOSS – GPL-like licence and multiplatform in order to enjoy continued development with no restrictions on the scientific community. The main strength of

open source software is that it allows the reuse of knowledge, and it can make it easier, by reducing programming time, to investigate new metrics or paradigms like Uncertainty or 3D metrics that are being considered for future research (McGarrigal et al., 2009). IAN and GRASS make this possible, and are good frameworks to develop on.

(3) Based on open standards promoting better integration with other existing or future projects. In the spatial domain, interoperability of software components is endorsed by the application of standards, with the Open Geospatial Consortium (OGC) being the main non-profit organization devoted to the elaboration of public and open specifications for geographic data and services interoperability. This means that we can use other open source projects to solve our needs, and there will be no significant problems when those projects make changes or improvements because they are based on known standards. The most recent projects reviewed here try to comply with this premise. In order to appreciate the relevance of this point look at http://www.opengeospatial.org/resource/products/ to check some popular products working with OGC standards.

(4) Possibility of extending the available data formats. As we have seen, a major concern in existing land-metrics software is data input, and readable formats. In order to be able to increase the accessible formats, a scalable application is required. Currently, many GIS programs use the GDAL/OGR library for this purpose, including different commercial apps (http://trac.osgeo.org/gdal/wiki/SoftwareUsingGdal).

(5) Extensible to all possible metrics. It is necessary to have a complete toolbox avoiding the use of more applications in order to reduce program training time.

(6) Usage of vector and raster formats depending only on the study requirements. Only PA4 gives the possibility of calculating both, vector and raster metrics. This is necessary to preserve data integrity and to compare results.

(7) Addition of new formulas and integration of future methodologies should be easy. Landscape metrics can be hundreds but only some of them are highly correlated. Due to the wide range of possible metrics the implementation of new formulas should be very easy. This is possible using OOP and provides the possibility of programming with many languages. As an example, GRASS is mainly programmed in Ansi-C which is a Low-level language (requires highest programming skills) and not OOP. On the other hand, IAN, written in Ruby is OOP and easy to use, due to this it can be a good possibility to develop.

(8) Customizable outputs and reports easily adaptable to our workflows. The outputs and reports that some programs return are generic and are often unnecessary or not exactly what is needed. The way to reduce unnecessary files or results is to give the possibility of calculating each metric directly, without intermediate inputs.

To achieve these goals a decision was taken to create an API which accomplishes these 8 points: It is the Land-metrics DIY (“do it yourself”).

2. Base technologies of Land-metrics DIY

Land-metrics DIY is a FOSS GPL licensed library which provides precise tools for creating specific applications for land-metrics calculation easily.

In this section we explain all the relevant decisions that have been taken to design Land-metrics DIY. Here all the technical details of the components and methodologies are not explained in depth, but a summary of the main issues we consider of interest for landscape and other researchers is indispensable. Of course, a further study of the following concepts is recommended in case of wanting to explore, use or develop the API.

2.1. Advantages of using the .NET Platform

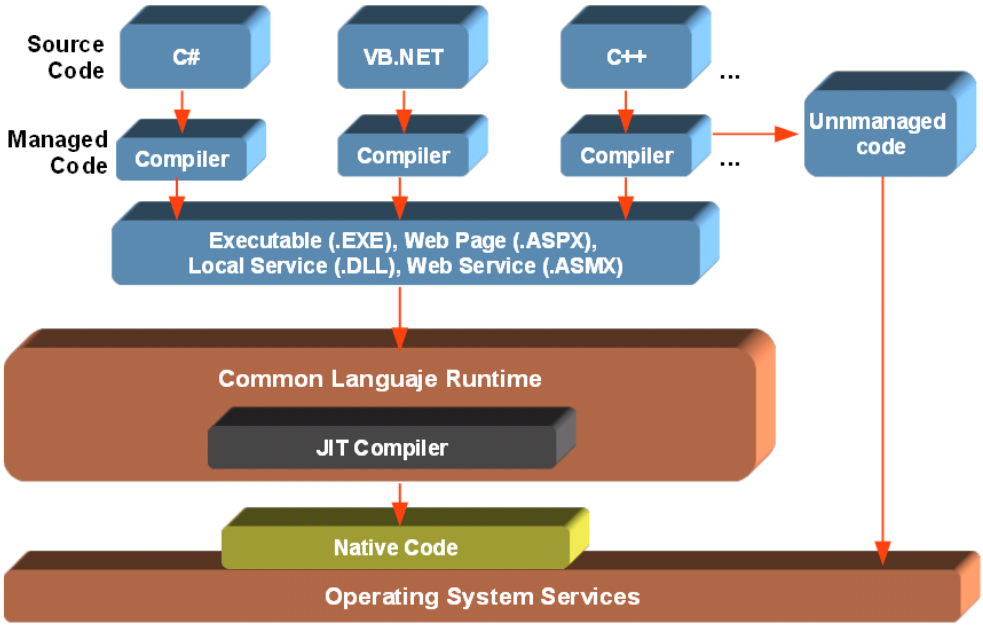
As stated before, in the development of our project it is especially interesting to achieve platform and programming language independency. Such a characteristic opens the project to almost any user and developer community. By allowing the use of any programming language, developer communities can make the most of our project minimizing development time and effort.

We have achieved multiplatform and multi-language support by implementing our API in the .NET development framework. Although it was originally designed by Microsoft for the Windows OS, the .NET framework was released as an open standard (ECMA-334-335, ISO/IEC 23271) making it available to third parties. Consequently, .NET can be implemented over any operating system. Using this standard, the Mono Project (http://www.mono-project.com/), a Novell initiative, has ported .NET platform to a still growing set of OS, such as GNU / Linux, FreeBSD, UNIX, Mac OS X, Solaris, and even Windows (with a different runtime to Microsoft's one).

"The .NET Framework is Microsoft's platform for building applications that have visually stunning user experiences, seamless and secure communication, and the ability to model a range of business processes. By providing you with a comprehensive and consistent programming model and a common set of APIs, the .NET Framework helps you to build applications that work the way you want, in the programming language you prefer, across software, services, and devices." (http://www.microsoft.com/net/).

The .NET execution model has many particularities that have to be explained (see **Figure 1**). When compiling a program, in a very similar way to the Java platform, an intermediate and platform independent language is generated. This intermediate language (IL) must be executed by an important piece of the .NET framework: the CLR (Common Language Runtime). This CLR acts in a similar way to the Java Virtual Machine allowing the code to be finally executed. This makes programming much easier thanks to the garbage collector, exception management and debugging features.

Figure 1: Execution model of the .NET platform

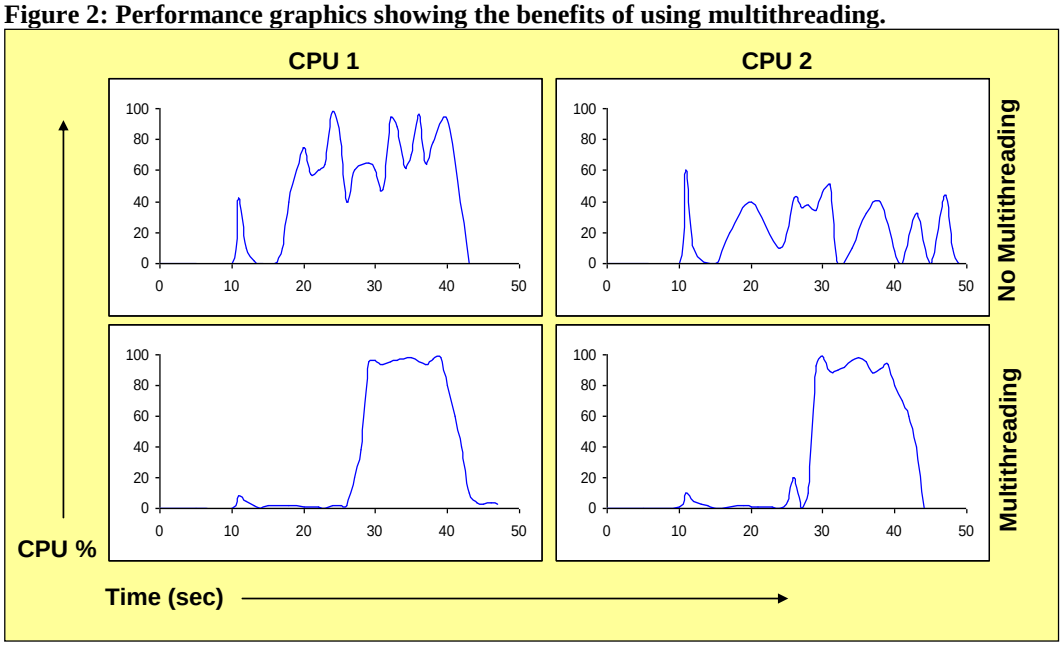


Nevertheless, an important difference with Java is that .NET is able to accept not only C# but any programming language as long as they follow the guidelines of the ECMA 335 known as the CLI (Common Language Infrastructure). That means that there currently exists a wide set of programming languages that can be used in order to create .NET applications (C++, VB.NET, F#, IronPython, IronRuby, Pascal, to name just a few). A complete list of currently supported languages can be seen at: http://en.wikipedia.org/wiki/List_of_CLI_languages. The multi-language nature of .NET allows mixing different programming languages in the same project.

As an additional part of the framework, .NET comes with a large collection of classes that allows the development of any type of application. From the basic and general Framework Class Library (FCL) in order to work with mathematical functions, XML information and other basic characteristics, to others more specific for database management (ADO.NET), web development (ASP.NET), outstanding graphic interfaces (WPF), and a large collection of possibilities.

As a modern Framework, .NET makes the most of computational performance. For instance, multithreading processing is a key resource for optimizing both vector and raster metrics calculation, and one that cannot be employed when programmers are tied to languages such as VB6, as is the case of PA4 and V-Late. In fact, not being able to implement multithreading can stop programmers from deploying solutions for large vector sets, which are the most CPU and memory consuming. In a simplified way, multithreading can be defined as the ability to execute simultaneous tasks making the most of the hardware configuration. For example, a dual core CPU is able to process two tasks at the same time, and this can be used to reduce by almost half the time in some geometric and raster operations. You can see an example in **Figure 2**. This graph shows differences in performance of a *CPU dual core (T5550)* when calculating the same patch core-metrics for a testing shapefile. Results reveal that by using multithreading, the processing time is halved and both processors work almost the same.

Figure 2: Performance graphics showing the benefits of using multithreading.



Among the different languages .NET allows to work with, **C#** has been chosen for our project as it is the most popular in the development of .NET applications. It is a simple object-oriented, modern, and general-purpose programming language and is also defined as standard (ECMA-334).

If the multi-language nature of .NET does not fit developers' expectations, the FOSS-GPL licence of our Land-metrics DIY project makes the conversion and adaptation of the code to other platforms such as Java, for example, straightforward.

The development of .NET applications is carried out using the Object-Oriented Programming (OOP) paradigm. Based on the concept of object, this paradigm resembles the human style of thinking in the programming world. Besides, OOP allows better code reuse, productivity and scalability. Particularly in our project, the addition of new metrics or other improvements can be easily carried out without introducing incompatibilities with previous versions and always allowing new features to be included for any development based on our API.

2.2. Approaching standards. The OGC Simple Feature Access (SFA)

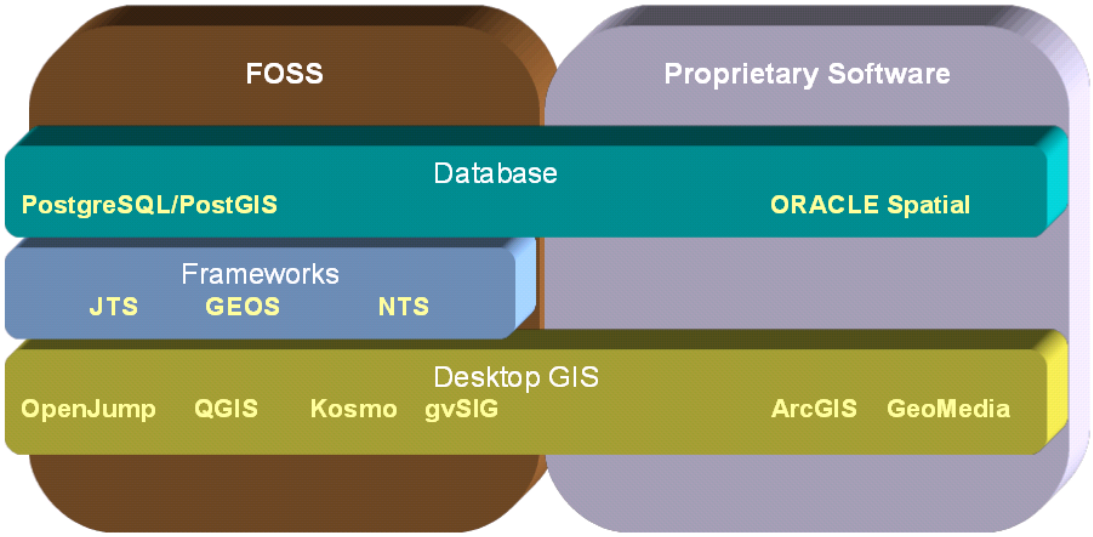
As has already been mentioned, we have adhered to open and well-known standards for the development of the Land-metrics DIY API. The most central standard used here is the OGS OpenGIS Simple Feature Access specification (SFA).

The SFA is an OpenGIS abstract specification that describes a feature model, and provides a data storing and an information access interface. These are the essential parts of a data model. Finally, Simple Features are geospatial features described using vector data elements such as points, lines and polygons.

Using these specifications the GIS software engineers are able to develop spatial applications that can manage simple geometries using different technologies. As can be

seen in **Figure3**, final applications can combine database servers, different frameworks designed using many programming languages and GIS desktop applications, without any dependence on software licences. In the OGC Webpage more solutions can be found that use SFA, but in **Figure 3** you can see the most popular.

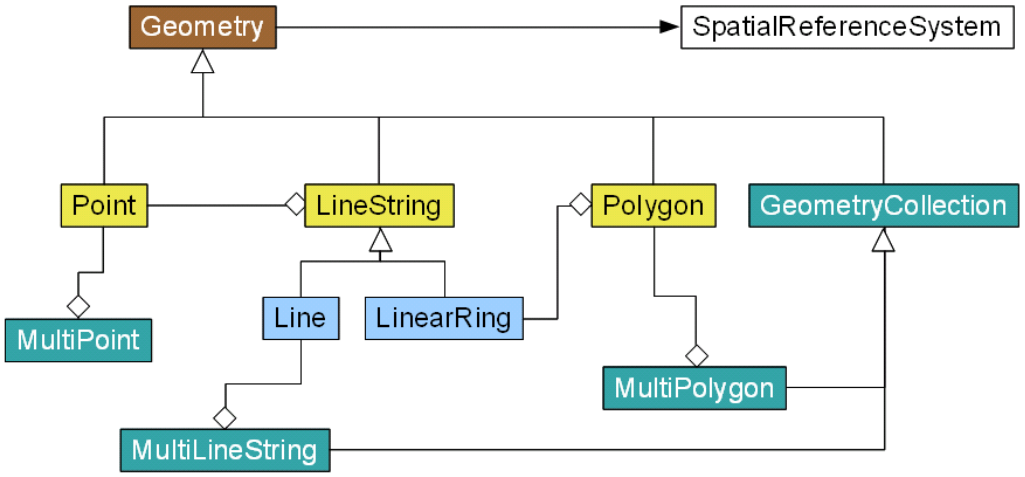
Figure 3: Interoperability using the SFA



The core of this specification is an Object Oriented Model which defines a neutral architecture to represent and analyze vector geographic features. SFA uses UML to explain its characteristics visually and at different scales. We can distinguish a data model and a functional model. In the next point, we explain more concepts about OO modelling and UML, but in order to work and to achieve a better understanding of SFA a deeper study of these concepts is required.

In **Figure 4**, we show a simplified UML class diagram that sums up many of the SFA data model details. This schema illustrates the main existing classes using rectangles and its relationships through arrows. A class is the definition of an object, and the different relationships are defined in UML using unlike heads to express aggregation (empty rhombus), dependence (full triangle) or inheritance (empty triangle).

Figure 4: Simplified SFA UML class diagram

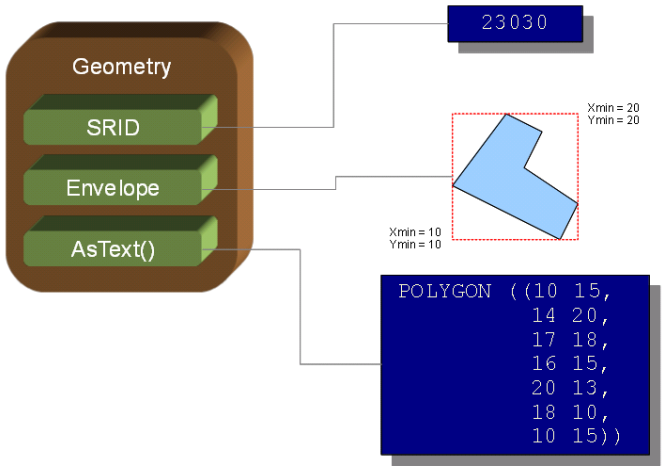


In summary, in the SFA data model the main base class is an abstract class called Geometry. Its main role is to provide a specification of the properties and methods common to any geometry. Among these properties the spatial reference system of coordinates is mandatory, and represented in turn by the SpatialReferenceSystem class. Three classes are derived from the Geometry class and represent the dimensionless (Point), unidimensional (LineString) and two-dimensional (Polygon) primitive geometries. Finally, the internal organization of the coordinates of polylines components (2 or more objects of type Point) and polygons (one or more objects of type LinearRing) as homogeneous collections of geometries (MultiPoint, MultiLineString , MultiPolygon) can be explained by aggregation relationships.

This data model is valuable for representing the landscape as patches using Polygons or exploring the landscape connectivity considering the nodes and edges of a network as Points and LineStrings.

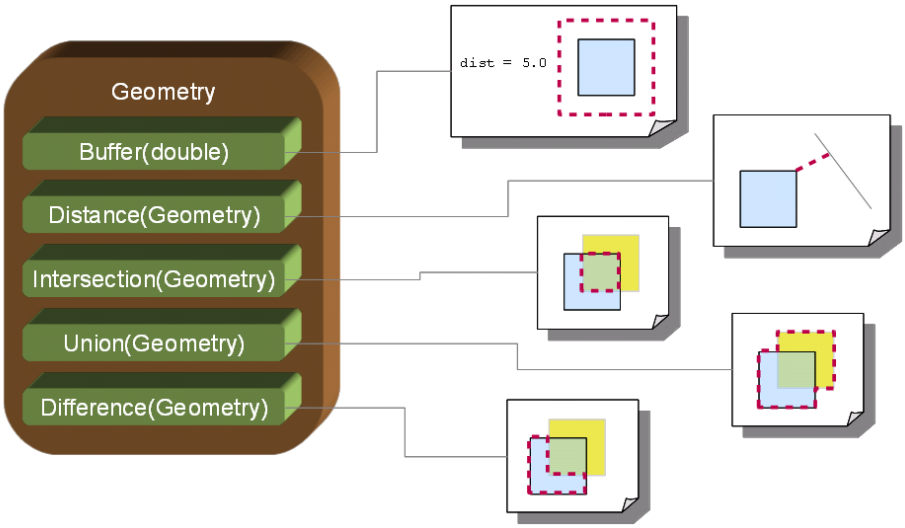
Once the SFA data model has been defined, we can explain the functional model which defines that each Geometry must include some necessary methods, some of them being immutable property accessors (**Figure 5**), and other more advanced and oriented to spatial analysis (**Figure 6**).

Figure 5: Geometry basic methods



Obviously, the advanced methods defined by the SFA are useful for calculating land-metrics. For example, the Buffer() method used with a negative depth of edge can be used to calculate the size of the Core Area of a Polygon.

Figure 6: Some Geometry advanced methods



2.3. Use of existing projects. "Not reinventing the wheel".

It seems absolutely true that the fastest project developed is that which is already done. Still using .NET and OOP the task of an API as complete as posed here would be enormous, or almost impossible due to the specificity of this knowledge area. Fortunately, nowadays the FOSS community is huge too and there are a lot of projects we could use to supply our lack of resources.

Table 2: Stats of some usable FOSS projects directly from .NET; Source: www.ohloh.net

Project	Types	Lines Of Code	Est. cost (US $)*
SharpMap	Data acces, spatial query, render	413.787	5.942.368
NetTopologySuite	2D spatial analysis	123.200	1.670.203
PostgreSQL + PostGIS	Spatial database	699.037	10.357.673
GeoAPI.NET	OGC/ISO, interoperability	-------	-------
GDAL	Raster access	640.817	9.691.503
Proj.NET	Projection Enghien	11.217	134.502

Because there are completely free FOSS projects for land metric calculations based on raster formats, we have decided to start the Land-metrics DIY by programming some vector formulas.

In **Table 2**, we can get a size approximation of some FOSS projects we could use from .NET. The methodology used to calculate these metrics is COCOMO II, which has the problem that in every project programmers try to reuse the code when possible, but it is evident that there is a lot of work already done. For the moment, only two FOSS

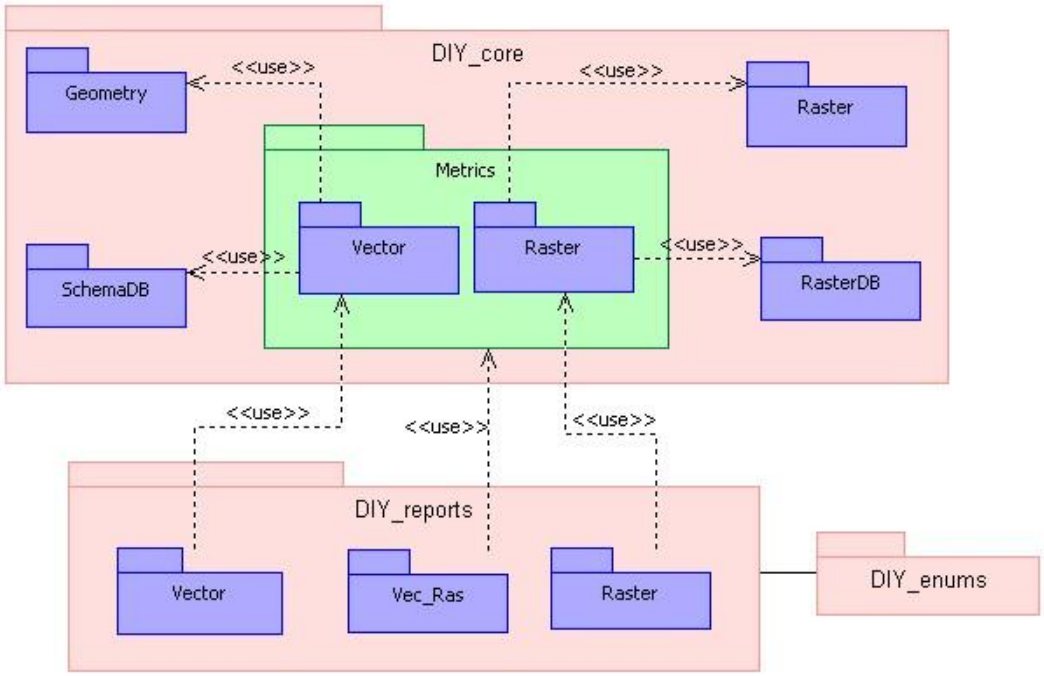
projects have been used to develop the prototype (Alfa 0.1) version of our API: NetTopologySuite (v. 1.7.3 Build 416) and GeoAPI.NET (1.1.0.0).

NTS is mainly a C#/NET porting of Java Topology Suite (JTS), a Java library for GIS operations, OpenGIS compliant (SFA). Both provide fundamental geometric functions with robust 2D spatial algorithms. NTS provides OGC standard geometry model implementation, read-write capabilities for standard vector GIS formats, some overlay functions (intersection, difference, union, symmetric difference), buffer, convex hull, area, perimeter and more. On the other hand, the GeoAPI.NET project provides a common framework based on standards to improve interoperability among .NET GIS projects, such as NTS.

3. General understanding of Land-metrics DIY.

In order to get a general understanding of how DIY works, we represent it as an UML package diagram. This diagram (**Figure 7**) shows the main namespaces in the land-metrics DIY project. A namespace is just a folder containing other folders or related classes. Finally, a class is the necessary code defining how an object of such class is created, its transitional states (properties) and its behaviour (methods).

Figure 7: UML package diagram of Land-metrics DIY



Following the diagram the raster/vector duality is appreciated, which is important because many metrics can be calculated in both ways (Wade et al., 2003).

In the centre, namespaces into "Metrics" are the most important because all the metric calculation classes are defined there. Also, in "DIY_core" there are 4 more namespaces, which contain the code for accessing the user GIS data files (i.e.: Shapefiles, GeoTiff or spatial databases). At the moment, only Shapefile format is readable, but more data formats will be incorporated before long. These classes allow some necessary information to be obtained like areas, perimeters, patch types, table fieldnames and any indispensable information to explore data and calculate land-metrics. As the diagram shows, in this version, when the user creates a "vector metric object" some classes in Geometry and SchemaDB are activated and use the appropriate tools from NTS and GeoAPI to get the necessary data.

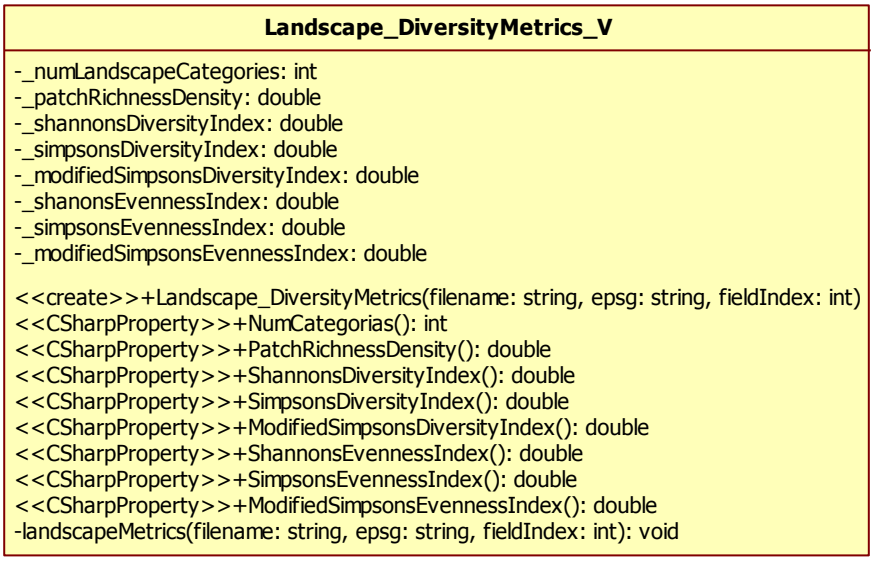
On the other hand, DIY_reports contains some tools to customize the output metrics list as needed. The list-objects receive the metrics calculated by the metrics namespace and list them as desired. There are some listing possibilities: (1) to get a complete list with all the metrics of the same category; (2) to get this list filtered with only the required metrics; (3) or combine lists from different categories but always at the same landscape level.

Of course, it is possible to calculate individual metrics without using the reports namespace.

Finally, in DIY_enums, there are enumerates, which are lists of terms that stop us from making typing errors (i.e. Lists with the metric names).

The steps for calculating all the metrics are always the same because all the classes in "metrics" follow the UML class schema in **Figure 8**. The first section of the schema defines the class name. By convention, each public class for metrics calculation in Landscape-metrics DYI is given a name composed of the landscape scale (patch-class-landscape for groups of metrics, and P, C or L in case of individual metrics), the metric name or the category metrics name. When possible, these names follow the Fragstats documentation. Finally, there is a letter that indicates the vector/raster option.

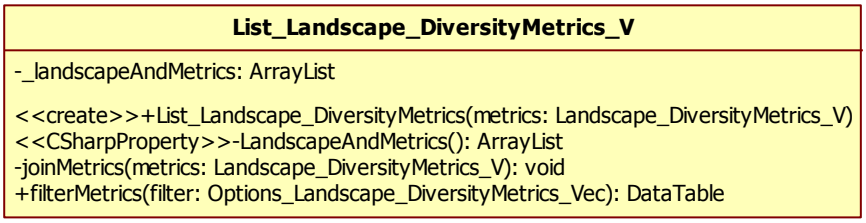
Figure 8: UML class schema for calculating a metrics category.



To understand the class schema, it is necessary to explain the concept of visibility. The API users will be able to use some tools from the dll, (the visible tools; +), but other tools will be invisible (-) to make it easier for the programmer. In the second section of the class schema there are one or more metric fields or attributes, always with private visibility (-). Those fields are queried using the public (+) methods in the third section. There will be a constructor (a public method with the same name as the class) that creates the landscape-metrics object and assigns values to metric fields (UML notation: <<create>>), and then it is possible to read the results through the corresponding public property accessors (<<C#Property>>). In this section there will always be a private method, which is actually in charge of calculating each metric. This way, the internal implementation of metrics calculation is encapsulated and isolated from the API's user application logic.

On the other hand, the classes in the reports namespace are well defined too. In **Figure 9**, you can see the UML schema that we can use to customize in the easiest way a report based on the metrics calculated following the schema in **Figure 8**. As we can appreciate, in any List class we will have only two public methods. The class constructor requires a Metrics object, and there is a public method called "filterMetrics" that will help us to create a customized "DataTable" just specifying the results we need to show or store.

Figure 9: UML class schema for customize some reports.



Also, into the reports namespace we will find some public and generic methods to work with the class reports (DataTables). Using these methods we could join two reports or sort the lists.

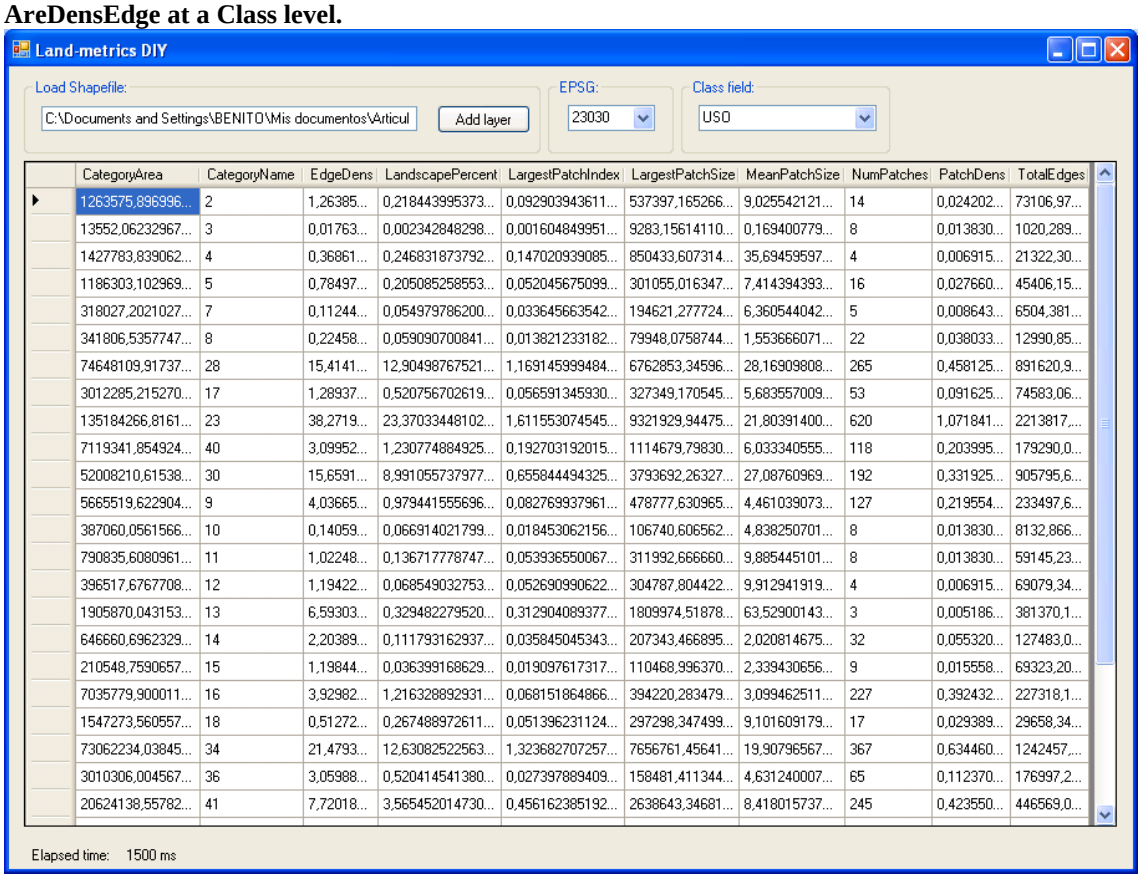
In this prototype we are only reporting lists or tables containing the results, but it is already planned to implement, for example, raster or vector graphical outputs. And this will be as easy to use as the example in **Figure 9**. Another issue is that thanks to .NET we can easily serialize our results to a XML file.

4. Results and Discussion

The main result of this research is a functional API for calculating landscape metrics directly from vector files.

The API prototype is able to calculate almost 40 different vector metrics (see Apendix C) from ESRI Shapefiles. Nevertheless, there will be a generic application for users that would not be interested in developing upon the library (**Figure 10**).

Figure 10: Sample User Interface (UI) distributed with the API that calculates all the available AreDensEdge at a Class level.



CategoryArea	CategoryName	EdgeDens	LandscapePercent	LargestPatchIndex	LargestPatchSize	MeanPatchSize	NumPatches	PatchDens	TotalEdges
1263575,896996...	2	1,26385...	0,218443995373...	0,092903943611...	537397,165266...	9,025542121...	14	0,024202...	73106,97...
13552,06232967...	3	0,01763...	0,002342848298...	0,001604849951...	9283,15614110...	0,169400779...	8	0,013830...	1020,289...
1427783,839062...	4	0,36861...	0,246831873792...	0,147020939085...	850433,607314...	35,69459597...	4	0,006915...	21322,30...
1186303,102969...	5	0,78497...	0,205085258553...	0,052045675099...	301055,016347...	7,414394393...	16	0,027660...	45406,15...
318027,2021027...	7	0,11244...	0,054979786200...	0,033645663542...	194621,277724...	6,360544042...	5	0,008643...	6504,381...
341806,5357747...	8	0,22458...	0,059090700841...	0,013821233182...	79948,0758744...	1,553666071...	22	0,038033...	12990,85...
74648109,91737...	28	15,4141...	12,90498767521...	1,169145999484...	6762853,34596...	28,16909808...	265	0,458125...	891620,9...
3012285,215270...	17	1,28937...	0,520756702619...	0,056591345930...	327349,170545...	5,683557009...	53	0,091625...	74583,06...
135184266,8161...	23	38,2719...	23,37033448102...	1,611553074545...	9321929,94475...	21,80391400...	620	1,071841...	2213817,...
7119341,854924...	40	3,09952...	1,230774884925...	0,192703192015...	1114679,79830...	6,033340555...	118	0,203995...	179290,0...
52008210,61538...	30	15,6591...	8,991055737977...	0,655844494325...	3793692,26327...	27,08760969...	192	0,331925...	905795,6...
5665519,622904...	9	4,03665...	0,979441555696...	0,082769937961...	478777,630965...	4,461039073...	127	0,219554...	233497,6...
387060,0561566...	10	0,14059...	0,066914021799...	0,018453062156...	106740,606562...	4,838250701...	8	0,013830...	8132,866...
790835,6080961...	11	1,02248...	0,136717778747...	0,053936550067...	311992,666660...	9,885445101...	8	0,013830...	59145,23...
396517,6767708...	12	1,19422...	0,068549032753...	0,052690990622...	304787,804422...	9,912941919...	4	0,006915...	69079,34...
1905870,043153...	13	6,59303...	0,329482279520...	0,312904089377...	1809974,51878...	63,52900143...	3	0,005186...	381370,1...
646660,6962329...	14	2,20389...	0,111793162937...	0,035845045343...	207343,466895...	2,020814675...	32	0,055320...	127483,0...
210548,7590657...	15	1,19844...	0,036399168629...	0,019097617317...	110468,996370...	2,339430656...	9	0,015558...	69323,20...
7035779,900011...	16	3,92982...	1,216328892931...	0,068151864866...	394220,283479...	3,099462511...	227	0,392432...	227318,1...
1547273,560557...	18	0,51272...	0,267488972611...	0,051396231124...	297298,347499...	9,101609179...	17	0,029389...	29658,34...
73062234,03845...	34	21,4793...	12,63082522563...	1,323682707257...	7656761,45641...	19,90796567...	367	0,634460...	1242457,...
3010306,004567...	36	3,05988...	0,520414541380...	0,027397889409...	158481,411344...	4,631240007...	65	0,112370...	176997,2...
20624138,55782...	41	7,72018...	3,565452014730...	0,456162385192...	2638643,34681...	8,418015737...	245	0,423550...	446569,0...

The Land-metrics DIY source code is publicly available for checkout in our own Subversion repository (http://www.gisandchips.org/svn/landmetrics_diy). The repository is organized in different folders: the "trunk" directory holds the main development branch, while the "tags" directory will hold subsequent functional releases either unstable or stable. There are two projects: the landmetrics_DIY API and landmetrics_DIY Visual. The second is a Windows Forms project consisting of a sample application of the API. **Figure 10** is a view of the Visual project.

In the "tags" folder you can see that the project actual release is a 0.1 version. In the near future, we have planned to achieve a beta version with some raster metrics calculations available and then we will port the complete project to an externally hosted sources repository (Google Code, SourceForge, etc.).

In the mean time, all those interested in Land-metrics DIY can visit the blog at http://www.gisandchips.org where short articles will appear explaining some issues about the use and development of the API, and its documentation too.

4.1. Usability of the API

Once the logic of the classes contained by the Metrics Namespace has been explained, we can see how easy would be to calculate some metrics. In this section we want to show how easy it is to use Land-metrics DIY in your own applications.

Then, we show some code examples. We must take into account some issues to understand them better:

(1) In C# the code lines end with a semicolon, while the lines starting with "//" are just comments for the readers.
(2) Parameter 1, "GIS filename", refers to the complete filename including the folder tree and extension of the filename. For the moment, the prototype only accepts ESRI shapefiles.
(3) Parameter 2, EPSG (European Petroleum Survey Group), requires an EPSG Spatial Reference System Identifier (SRID), which is a value to identify a spatial coordinate system (http://en.wikipedia.org/wiki/EPSG).
(4) Parameter 3 is the position in a related data table of the field used to identify the patches.

See the following C# code, there are two possibilities available: (1) calculate just one metric (Figure 11; A, B) or (2) calculate all the metrics related to the same class and at the same landscape level (Figure 11; C).

Figure 11: This C# code calculates the area and perimeter metrics at a patch level, and all the AreaDensEdge metrics at a patch level, using the vector formula.

A)
```
//We calculate only the area at the patch level
P_Area_V areas = new P_Area_V(GIS filename, EPSG, fieldIndex);
```

B)
```
//We calculate only the Perimeter at the patch level
P_Perim_V perims = new P_Perim_V(GIS filename, EPSG, fieldIndex);
```

C)
```
//We calculate all the metrics in this class
Patch_AreaDensEdge_V metrics = new Patch_AreaDensEdge_V(GIS filename,
EPSG, fieldIndex);
```

Once the metrics are calculated, we will want to customize our results by merging some metrics results into the same table (Figure 12; A), or you will need a subset of the metrics of the specific metrics class (Figure 12; B).

Figure 12: This C# code merges 2 metrics at a patch level, or creates a personalized output using the vector formula.

A)
```
//We merge the areas and perimeters calculated in code example Figure
11.A and 11.B
Join_Datatables.JoinTables(areas.C_Names_Areas_V,
perims.C_Names_Perimeters_V,"ID");
```

B)
```
//We create a tailored list from the metrics calculated in Figure 11.C
List_Patch_AreaDensEdge list = new List_Patch_AreaDensEdge(metrics);
list.filterMetrics(Options_Patch_AreaDensEdge_V.PatchName|Options_Patc
h_AreaDensEdge_V.PatchArea);
```

4.2. Program using other languages.

As we stated before, thanks to the .NET Framework, it is possible to use or program this API using different programming languages. In order to show a few examples, we have written the code example in **Figure 11-A** to calculate the area at the patch level in other 3 other programming languages: IronPython, IronRuby and VB.NET.

Figure 13: This code calculates patch areas, using different programming languages.

```
//C#
P_Area_vec pA = new P_Area_vec(GIS filename, EPSG, fieldIndex);

//VB.NET
Dim pA As New P_Area_V(GIS filename, EPSG, fieldIndex)

//IronRuby
pA = P_Area_V.new(GIS filename, EPSG, fieldIndex)


//IronPython
pA = P_Area_V.new(GIS filename, EPSG, fieldIndex)
```

In the code examples in **Figure 13**, we can compare the different syntax of 4 programming languages that fit .NET. All of them are very similar, but any user could have preferences or an acquired programming background which make them prefer a particular language.

4.3. Testing Land-metrics DIY

At this point, we would like to list the main issues that make Land-metrics DIY a good option for working with landscape metrics. The best way to do this is to see all the tools working in a test.

First of all, we calculated all the available metrics in our API by hand, only when possible, and compared the results with those offered by the rest of the best known programs referred to in this paper. All the results were correct.

The only problem was to compare the results obtained using the vector metrics with those returned by using raster formulas. The pixel size problem made it possible to compare only a few metrics. In spite of this problem, we can compare the packages which perform vector calculations (V-Late and Patch Analyst 4). Nevertheless, we have taken into account the results obtained with Fragstats and PA4-raster for a rasterized file of 1x1 meters per pixel. Some of the results were similar.

The calculations of the core-area metrics are the most highly demanding in the case of the vector metrics implemented in this article. At the moment we tried to calculate the core-area metrics using V-Late, none of the computers used (**Apendix B.1 and B.2**) could perform the task, but PA4 did. In fact, the API prototype still does not apply multithreading and is not optimized but despite this Land-metrics DIY was faster and calculated more metrics from this group than PA4.

The Land-metrics DIY calculates many metrics that are not available in the ArcMap extensions like some diversity or core indexes (see **Appendix C**). However, the main question is that it is relatively easy to program new metrics in the API, following the schemas provided here, but this is not possible in the rest of vector software.

At this moment, the API works very fast, at least when we compare it with the other solutions. One of the reasons is that the API enables results to be visualized before saving them to a file, avoiding unnecessary writing tasks.

Since the API has been designed to provide metric calculations the creation of reports has not been one of the main interests. However, the .Net Framework makes it possible to export the results to a text file or XML in a very easy way. There are some examples in the code of the "Visual" project.

5. Conclusions

Nowadays, there are many existing possibilities to facilitate the creation of customized software. This is also true in the case of spatial software. Using a friendly framework like .Net, well defined and distributed standards like the OpenGIS and some FOSS projects oriented to GIS make it easy to create powerful tools with less programming effort and skills. Knowing the resources we have is basic for scientists. This allows us to avoid some traditional problems when creating new software.

In this day and age, there are many researchers advancing the needs for new landscape metrics (McGarigal et al., 2009) or new methodologies (Rempel and Kaufman, 2008; Saura and Torné, 2009), and this will result in the development of new software. It is essential to be planners at this moment to design the best possible tools, avoiding the inconveniences of the past. We conclude that the project set out here fulfils the requirements indicated in the introduction of this paper, solving those traditional problems and providing a good framework for further developments.

Due to the interests of the authors, this is a FOSS project that is very alive and further developments are being worked on now. The project is in its initial stages but during the following months we will provide access to a wide range of data formats, raster capabilities, and we will try to complete the list of metrics available as much as possible, using both raster and vector formulas. On the other hand, we have to improve the reports namespace making the customization of results easier and providing when needed the possibility of saving the results in GIS files.

Acknowledgements

This paper is partially supported by the FPU - Doctoral Research Scholarship program of the "Ministerio de Educación de España". (2007-2011). Moreover we would like to thank all the FOSS community, and in particular to Diego Guidi (NTS main developer) for their interesting and important work. Of course, we also thank the OGC and all

FOSS projects related to spatial information because they suppose a strategic support to the further development of our idea and other possible GIS projects.

Appendix A. - List of acronyms (which are not sufficiently explained in the text):

ANSI — C — a standarization of the C programming language
API — Application Programming Interface
ArcGIS — a set of GIS programs produced by ESRI
ArcMap — one of the main ArcGIS components
ArcObjects — ESRI environment for programming GIS applications
C++ — low level and multiparadigm programming language
COCOMO II — COnstructive COst MOdel — Mathematic model for estimating the costs of a software development
ECMA — European Computers Manufacturers Asociation — Now, ECMA International; an organization that designs and promotes the correct use of standards, and publishes them in the public domain
EPSG SRID — European Petroleum Survey Group Spatial Reference Identifier
ESRI — Environmental Systems Research Institute
FOSS — Free and Open Source Software
GDAL — Geospatial Data Abstraction Library — is a library for reading and writing geospatial data formats
GIS — Geographical Information System — is any system that captures, stores, analyzes, manages, and presents data that is linked to location
GNU — 'GNU's Not Unix', a recursive acronym— a caricature of the wildebeest/gnu is often used as logo for the GNU project
GPL (GNU GPL) — General Public License
GRASS — Geographic Resources Analysis Support System
OOP — Object-Oriented Programming
OpenGIS® — a Registered Trademark of the Open Geospatial Consortium, Inc (OGC). It is associated with the Standards and documents produced by the OGC
OSGeo — Open Source Geospatial Foundation — "not-for-profit" organization whose mission is to support and promote the collaborative development of open geospatial technologies and data
Phyton — high level programming language for general purposes
r.le — GRASS module which provides many quantitative measures of landscape (2001)
r.li — another Grass module like r.le, but newer (2008)
RUBY — a dynamic, reflective, general purpose and object-oriented programming language
Shapefile — GIS file-format property of ESRI but today opened and used like a standard
Source Code — refers to the original text form of a computer program
UML — Unified Modelling Language — a standardized general-purpose modelling language for software engineering. Includes several techniques to create visual models for a software design
VB 6 — Visual Basic 6 — easy event-oriented programming language, previous to the .NET Framework
XML — eXtensible Markup Language — a free and open standard specification that contains a set of rules for encoding documents electronically

Appendix B. - Materials used to test Land-metrics DIY

Here the characteristics of the materials used for testing the API are explained.

B.1. – Personal computer
Processor: Intel Core2
Memory 1 (RAM): 2 Gb
Memory 2 (Free hard disc): 55 Gb
Operating system: Microsoft Windows XP

B.2. –Laptop
Processor: Intel Core2 – Duo T5550
Memory 1 (RAM): 2 Gb
Memory 2 (Free hard disc):32 Gb
Operating system: Windows XP, Linux Ubuntu 8.10

B.3. – Shapefile with a land use classification
Memory size: 8.81 Mb
Number of polygons: 3675
Number of classes: 22
Spatial Reference (EPSG): 23030

Appendix C. - Land-metrics DIY available metrics list, and smart comparison with 2 vector-based applications:

X	Test failure
-	Not implemented/Indirect
●	Matches
○	Aprox. Matches

	DIY Names	Short Names	V-Late	PA4	DIY
Patch	PatchArea	AREA	●	●	●
	PatchEdge	PERIM	●	●	●
	CoreArea	CA	**X**	●	●
	CoreAreaIndex	CAI	**X**	-	●
	NumCoreAreas	NCORE	**X**	-	●
	PatchFractalDimension	PAFRAC	●	-	●
	PatchPerimeterAreaRatio	PARA	●	-	●
Class	CategoryArea	CA	●	●	●
	EdgeDensPerCategory	ED	-	●	●

	LandscapePercentPerCategory	PLAND	-	-	●
	LargestPatchIndexPerCategory	LPI	-	-	●
	MeanPatchSizePerCategory	MPS	-	●	●
	NumPatchesPerCategory	NP	●	●	●
	PatchDensPerCategory	PD	-	-	●
	TotalEdgesPerCategory	TE	●	●	●
	CategoryCoreArea	TCA	**X**	●	●
	CoreAreaPercentOfLandscape	CPLAND	-	-	●
	DisjunctCoreAreaDensity	DCAD	-	-	●
	NumDisjunctCoreAreas	NDCA	**X**	-	●
Landscape	LandscapeArea	TA	●	●	●
	LandscapeEdge	TE	●	●	●
	LandscapeEdgeDensity	ED	●	●	●
	LandscapeLargestPatchIndex	LPI	-	-	●
	LandscapePatchesDensity	PD	-	-	●
	LandscapeShapeIndex	LSI	-	-	●
	NumLandscapePatches	NP	●	●	●
	DisjunctCoreAreaDensity	DCAD	-	-	●
	NumberOfDisjunctCoreAreas	NDCA	-	○	●
	TotalCoreArea	TCA	**X**	●	●
	ModifiedSimpsonsDiversityIndex	MSIDI	-	-	●
	ModifiedSimpsonsEvennessIndex	MSIEI	-	-	●
	NumLandscapeCategories	PR	●	-	●
	PatchRichnessDensity	PRD	-	-	●
	ShannonsDiversityIndex	SDI	●	●	●
	ShannonsEvennessIndex	SEI	●	●	●
	SimpsonsDiversityIndex	SIDI	-	-	●
	SimpsonsEvennessIndex	SIEI	-	-	●

References

- Baker, W.L., 2001. The r.le programs: a set of GRASS programs for the quantitative analysis of landscape structure. Department of Geography. University of Wyoming.
- Baker, W.L., Cai, Y., 1992. The r.le programs for multiscale analysis of landscape structure using the GRASS geographical information system. Landscape Ecology 7, 291–302.

- Brennan, S.P. and Schnell, G.D., 2005. Relationship between bird abundances and landscape characteristics: the influence of scale. Environmental Monitoring and Assessment 105, 209–228.
- Calenge, C., Maillard D., Fournier, P. and Fouque, C., 2004. Efficiency of spreading maize in the gariques to reduce wild boar (*Sus scrofa*) damage to Mediterranean vineyards. European Journal of Wildlife Research 50,112–120.
- Cartwright, J. C., 1991. IDRISI-Spatial Analysis at a Modest Price. GIS World, Vol. 4, No. 9, p. 96-99.
- Coulson, T., Catchpole, E.A., Albon, S.D., Morgan, B.J.T., Pemberton, J.M., Clutton-Brock, T.H., Crawley, M.J. and Grenfell, B.T., 2001. Age, sex, density, winter weather, and population crashes in Soay sheep. Science 292, 1528-1531.
- Garel, M., Cugnasse, J.M., Loison, A., Gaillard, J.M., Vuiton, C. and Maillard, D., 2005. Monitoring the abundance of mouflon in South France. European Journal of Wildlife Research 51, 69–76.
- Guillemain, M., Mondain-Monval, J.Y., Weissenbacher, E., Brochet, A.L. and Olivier, A., 2008. Hunting bag and distance from nearest day-roost in Camargue ducks. Wildlife Biology 14(3), 379-385.
- Gustafson, E.J., 1998. Quantifying Landscape Spatial Pattern: What Is the State of the Art? Ecosystems 1, 143–156.
- Hebeisen, C., Fattebert, J., Baubet, E. and Fischer, C., 2008. Estimating wild boar (*Sus scrofa*) abundance and density using capture–resights in Canton of Geneva, Switzerland. European Journal of Wildlife Research 54, 391–401.
- Jiménez-García, D., Martínez-Pérez, J.E. and Peiró, V., 2006. Relationship between game species and landscape structure in the SE of Spain.Wildlife Biology in Practise 2(2), 48-62.
- Kaden, V., Hänel, A., Renner, Ch. and Gossger, K., 2005. Oral immunisation of wild boar against classical swine fever in Baden-Württemberg: development of the seroprevalences based on the hunting bag. European Journal of Wildlife Research 51, 101–107.
- Lang, S., Tiede, D., 2003. vLATE Extension für ArcGIS - vektorbasiertes Tool zur quantitativen Landschaftsstrukturanalyse, ESRI Anwenderkonferenz 2003 Innsbruck. CDROM.
- Le Pichon, C., Gorges, G., Baudry, J., Goreaud, F. and Boët, P., 2009. Spatial metrics and methods for riverscapes: quantifying variability in riverine fish habitat patterns. Environmetrics 20, 512–526.
- Lesschen, J.P., Cammeraat, L.H., Kooijman, A.M., Van Wesemael, B., 2008. Development of spatial heterogeneity in vegetation and soil properties after land abandonment in a semi-arid ecosystem. Journal of arid environments 72(11), 2082-2092.
- Lin, Y.P., Hong N.M., Wu, P.J., Wu, Ch.F. and Vergurg, P.H., 2007. Impacts of land use change scenarios on hydrology and land use patterns in the Wu-Tu watershed in Northern Taiwan. Landscape and Urban Planning 80(1-2), 111-126.
- Longley, P.A., Goodchild, M.F., Maguire, D.J. and Rhind, D.W., 2005. Geographic information systems and science, 2nd ed. Wiley, Chichester.
- Matthew F. B., Jess T.C. and Mark. W.J., 2009. Landscape metrics indicate differences in patterns and dominant controls of ribbon forests in the Rocky Mountains, USA. Applied Vegetation Science 12 (2), 237-249.

- McGarigal, K. and Marks, B.J., 1995. FRAGSTATS: spatial pattern analysis program for quantifying landscape structure. Portland (OR): USDA Forest Service, Pacific Northwest Research Station, General Technical Report PNW-GTR-351.
- McGarigal, K., Cushman, S.A., Neel, M.C. and Ene, E., 2002. FRAGSTATS: Spatial Pattern Analysis Program for Categorical Maps. Computer software program produced by the authors at the University of Massachusetts, Amherst. Available at the following web site: www.umass.edu/landeco/research/fragstats/fragstats.html
- McGarigal, K., Tagil, S. and Cushman, S.A., 2009. Surface metrics: an alternative to patch metrics for the quantification of landscape structure. Landscape Ecology 24, 433–450.
- Monzón, A. and Bento, P., 2004. An analysis of the hunting pressure on wild boar (*Sus scrofa*) in the Trás-Os-Montes Region of Northern Portugal. Galemys, 16, 253-262.
- Nadal, J., 2001. Global sex and age ratios in declining populations of red-legged partridges (*Alectoris rufa*) in the Province of Huesca (Spain). Game-and-Wildlife-Science 18(3-4), 483-494.
- Porta, C., Spano, L.D., 2008. The r.li module of the GRASS geographical information system. Website: http://grass.itc.it/grass63/manuals/html63_user/r.li.html
- Rempel, R.S., 2008. Patch analyst 4 - history. Available from: http://flash.lakeheadu.ca/~rrempel/patch/whats_new.html.
- Rico, M. and Torrente, J. P., 2000. Caza y rarificación del lobo en España: Investigación histórica y conclusiones biológicas. Galemys 12, 163-179.
- Schropfer, R., Bodenstein, C. and Seebass, C., 2000. A predator-prey-correlation between the European polecat *Mustela putorius* L., 1758 and the wild rabbit *Oryctolagus cuniculus* L., 1758. Zeitschrift Fur Jagdwissenschaft 46(1), 1-13.
- Hoechstetter, S. and Walz, U., 2009. 3D-metrics in Landscape Ecology – Methods and examples of use. European IALE Conference 2009, Salzburg (Austria).
- Simoniello, T., Carone, M.T., Coppola, R., D'Emilio, M., Grippa, A., Landfredi, M., Liberti, M. and Macchiato, M., 2006. Preliminary study to monitor land degradation phenomena though landscape metrics. Proceedings of the second Workshop of the EARSeL SIG on Land use and Land cover. Center for remote sensing of Land surfaces. Bonn.
- Steiniger, S. and Hay, G. J., 2009. Free and open source geographic information tools for Landscape Ecology. Ecological Informatics 4 (4), 183-195.
- Tsachalidis, E.P. and Hadjisterkotis, E., 2008. Wild boar hunting and socioeconomic trends in Northern Greece, 1993–2002. European Journal of Wildlife Research 54, 643–649.
- Turner, M.G., Gardner, R.H., O'Neill, R.V., 2001. Landscape Ecology in theory and practice: pattern and processes. Springer, New-York.
- Uuemaa, E., Antrop, M., Roosaare, J., Marja, R. and ¨Mander, U., 2009. Landscape Metrics and Indices: An Overview of Their Use in Landscape Research. Living Rev. Landscape Research 3 (1). [Online Article]: cited [2-10-2009]. http://www.livingreviews.org/lrlr-2009-1
- Vargas, J.M., Guerrero, J.C., Farfán, M.A., Barbosa, A.M. and Real, R., 2006. Land use and environmental factors affecting red-legged partridge (*Alectoris*

rufa) hunting yields in southern Spain. European Journal of Wildlife Research 52, 188–195.

- Visconti, P. and Elkin, Ch., 2009. Using connectivity metrics in conservation planning – when does habitat quality matter? Diversity and Distributions 15, 602–612.

- Visconti, P. and Elkin, Ch., 2009. Using connectivity metrics in conservation planning – when does habitat quality matter? Diversity and Distributions 15, 602–612.

- Wade, T.G., Wickham, J.D., Nash, M.S., Neale, A.C., Riitters, K.H. and Jones, K.B., 2003. A Comparison of Vector and Raster GIS Methods for Calculating Landscape Metrics Used in Environmental Assessments. Photogrammetric Engineering & Remote Sensing, 69(12),1399-1405.
- Wagner, H.H. and Fortin, M.J., 2005. Spatial analysis of landscapes: concepts and statistics. Ecology 86(8), 1975-1987.
- Wang, W.C., Yang, .J.L., Lin, Y.Y. and Taichung, T., 2008. Proceedings of the academic track of the 2008 Free and Open Source Software for Geospatial (FOSS4G) Conference, incorporating the GISSA 2008 Conference 29. Cape Town, South Africa. ISBN 978-0-620-42117-1
- Wu, J., 2006. Cross-disciplinarity, Landscape Ecology, and sustainability science. Landscape Ecology 21,1-4.