Jul 1st, 12:00 AM

Free and Open Source Geospatial Tools for Environmental Modeling and Management

Ari Jolma

Daniel P. Ames

N. Horning

M. Neteler

Aaron Racicot

See next page for additional authors

Presenter/Author Information
Ari Jolma, Daniel P. Ames, N. Horning, M. Neteler, Aaron Racicot, and T. Sutton

Free and Open Source Geospatial Tools for Environmental Modeling and Management

A. Jolma[a] **, D.P. Ames**[b]**, N. Horning**[c]**, M. Neteler**[d]**, A. Racicot**[e] **and T. Sutton**[f]

[a] *Helsinki University of Technology, Espoo, Finland, ari.jolma@tkk.fi*

[b] *Geographic Information Sciences Lab, Idaho State University, Idaho Falls, ID, USA*

[c] *Center for Biodiversity and Conservation, American Museum of Natural History, New York, NY, USA*

[d] *Center for Scientific and Technologic Research, The Trentino Cultural Institute, Trento, Italy*

[e] *Ecotrust, Portland, OR, USA*

[f] *BiodiversityWorld Project*

Abstract: Geospatial software tools (GIS) are used for creating, viewing, managing, analyzing, and utilizing geospatial data. Geospatial data can include socio-economic, environmental, geophysical, and technical data about the Earth and societal infrastructure and it is pivotal in environmental modeling and management (EMM). Desktop, web-based, and embedded geospatial tools and systems have become an essential part of EMM. Environmental simulation models often require pre- or post-processing of geospatial data, or they can be tightly linked to a GIS, using it as a graphical user interface (GUI). Many local, regional, national, and international efforts are underway to create geospatial data infrastructures and tools for viewing and using geospatial data. When environmental attribute data is linked to these infrastructures, powerful tools for environmental management are instantly created. The growing culture of free and open source software (FOSS) provides an alternative approach to software development also in the field of GIS (FOSS4G). For a systematic look at FOSS4G for EMM platforms, software stacks, and EMM workflows need to be analyzed. Platform is a service abstraction on which software stacks are built. A software stack for FOSS4G comprises system software, data processing tools, data serving tools, user interface tools, and end-user applications. Digital map creation, support for numerical modeling, and geospatial information systems are main areas of use for FOSS4G in EMM. The dividing line between FOSS and proprietary software is fuzzy, partly because it is in the interest of developers of proprietary software to make it fuzzy and partly because the end-users are getting reluctant to buy software. In the FOSS world the barriers to interoperability are low and thus the software stack tends to be thicker than in the proprietary platform. The FOSS4G world thrives on the evolution of software stacks and platforms. Our examples show that it is possible to build software stacks from current FOSS4G to support EMM workflows. In the examples we mention for example how a particular funding agency has chosen FOSS4G solutions because of the opportunities to redistribute resulting modeling tools freely to end-users and to support general goals of openness and transparency with respect to modeling tools.

Keywords: Free software; Open Source Software; Geospatial software; Geographic Information Systems; Environmental modeling and management

1. INTRODUCTION

Geospatial software tools are used for creating, viewing, managing, analyzing, and utilizing geospatial data. An interoperable collection of such tools may be called a Geographic Information System (GIS)[1]. And, the development of such

[1] We assume in this sentence a broad definition for a GIS. It is also possible to assume a more restricted definition and consider GIS separate from

tools and the study of their application to real-world problem solving may be called Geographic Information Science (GISci). Geospatial data can include socio-economic, environmental, geophysical, and technical data about the Earth and societal infrastructure. The common characteristic of all geospatial data is the presence of a spatial component: objects are tied to actual places and locations, referenced by geospatial coordinates.

Geospatial objects can encapsulate environmental attributes and link to each other in models of environmental networks. Geospatial objects can be temporally static or dynamic; they can represent data at a variety of spatial scales and resolutions; and they can be very simple (e.g. points) or complex (e.g. 3-D triangulated irregular networks). Geospatial data and tools can be used to address and answer many disparate types of questions in most every field of study. For these and other reasons, a large number of tools have been developed for working with geospatial data. One result of the proliferation of such tools is a coincident proliferation of a large number of storage formats which have been specifically designed for geospatial data. While many of these formats are open, i.e., their specifications have been published, others are proprietary and do not support data interoperability between tools.

Desktop, web-based, and embedded geospatial tools and systems have become an essential part of environmental modeling and management (EMM). With the incorporation of geospatial data sets these systems allow new and unique views into the environment. This is particularly true in the case of Internet-based tools that provide wide and relatively easy access to large geospatial data sets and useful analyses (e.g. driving directions, environmental data summaries, etc.) Many local, regional, national, and international efforts are underway to create geospatial data infrastructures and tools for viewing and using geospatial data and have put geospatial technology in the international spotlight. In many cases, these efforts build on free and open source tools and open, published data formats creating new challenges and opportunities for the EMM software community.

The GIS industry is growing rapidly, constituting a $2.02 billion international market in 2004 [Daratech, 2004]. Many very successful commercial companies have developed over the past 25 years, built on the model of delivering proprietary GIS solutions and selling software licensing. Many university curricula are taught around these software packages and many professional modelers, managers, and consultants depend on them in their work. We hypothesize here that this business model has resulted in a closed and monolithic structure in GIS software and its development, and that this has resulted in reduced interoperability, software transparency, and data transferability.

The growing culture of free[2] and open source software (FOSS) provides an alternative approach to software development also in the field of GIS. FOSS does not necessarily sacrifice business aspects of software development and application; there are several successful companies built around FOSS and companies are using FOSS components to a substantial degree [Bonnici, 2006]. Indeed, the FOSS model of software development has produced a new breed of software and associated business models. Certainly there are many shinning examples of FOSS successes such as the Linux/GNU operating system and its many brands. The community that develops FOSS for geoinformatics (FOSS4G) has recently gained momentum[3], making it a more viable option for environmental modelers and managers. A survey of FOSS4G has been presented by Ramsey [2005]. Published formal comparisons between FOSS4G and proprietary solutions are not common. Wikstrøm and Tveite [2005] compared PostGIS and MapServer favorably against ArcSDE and ArcIMS.

The relationship between environmental models (as software), which re-create a simplification of events in nature, and geospatial data (as stored in computers) has been studied for several decades. The initial phase of this research culminated in a series of conferences such as initial HydroGIS [Kovar and Nachtnebel, 1993] and initial GIS/EM [Goodchild et al, 1993], which presented the then state-of-the-art in development and application of geospatial tools in EMM. One result of these efforts has been the birth of new disciplines of hy-

e.g., remote sensing image analysis tools and CAD tools for geospatial data.

[2] We mainly use the term "free" to refer to the freedoms [Anonymous, 2005] the developer and user has with using the software or data, not to the price tag. Many tools are free to use or download, and they are sometimes useful, but the distinction is important to note.

[3] We refer to the founding of OSGeo (www.osgeo.org) and to the successes of the Open Source Geospatial conferences, e.g., OSG'05: http://mapserver.gis.umn.edu/community/conferences/MUM3/.

droinformatics and geoinformatics. At the same time geocomputation has also emerged as a discipline, focusing on the study of geospatial phenomena using computers. Despite these efforts, the problem of efficiently connecting environmental models and geospatial tools still exists and is an active area of research.

In this paper we examine the current state and future prospects for free and open source geospatial tools in EMM. We will analyze the current enabling technologies and tools along with future directions. Analysis of the role of the community and of the community process is left for another paper in the same workshop [Gross et al. in prep.]. We also do not consider problems related to intellectual property, such as licensing issues. We restrict our description and analysis to tools, which give modelers and managers the freedom of adopting, using, learning, and extending the tools, i.e., they can be considered FOSS[4]. The description reflects our current understanding of what are or will be the key building blocks of geospatially aware information systems for environmental problem solving. Links to web pages of the various software tools mentioned in this paper are not always given since they are usually very easy to find using Internet search tools. The discussed FOSS4G tools are only a representative set of what is available. Portals, such as freegis.org, attempt to maintain comprehensive lists.

For a systematic look at FOSS4G we divide our analysis in three main parts:

(i) Platforms – Platforms are defined as the media used to deliver FOSS4G solutions. We mainly examine and compare the desktop and the web as platforms. Despite this rather simplistic approach, a platform is a very versatile, interesting, and useful concept when software solutions are analyzed. Each platform offers a unique set of functionality and opportunity when applied to EMM.

(ii) Software Stack – Each platform has its own specific needs and available tools for building a working geospatial software stack. We will examine various software stacks for these two platforms and discuss functionality as well as interoperability. This should lead to a better understanding what services these tools can provide to environmental managers. Note that in this paper, we focus on tools, which are within the more restricted or traditional definition of a GIS, i.e., tools that are not necessarily used for remote sensing or CAD applications.

(iii) Workflows – Each platform and software stack is particularly suited for specific workflows. We will describe some common workflows and show FOSS4G solutions for them. Analyzing typical "use cases" (software usage scenarios) by diverse user types is intended to show the strengths and weaknesses of the current set of FOSS4G tools available and shed light on future opportunities for improvement.

2. PLATFORMS

An important trend in computing is the increased availability of new platforms: e.g. different versions of operating systems, different combinations of hardware, and different client-server protocols for delivering tools to users. As software becomes increasingly important in everyday life, more people use software, and more tasks are taken care of by software, the number of existing platforms grows. The concept has been popularized most visibly by Ray Ozzie, CTO of Microsoft and key developer of Lotus Notes, who has defined "platform" in his weblog as "a relevant and ubiquitous common service abstraction"[5].

The benefit of a platform to a user is a common look and feel and an interoperable set of functionality. The benefit of a platform to a developer is the ready availability of interoperable tools and services, e.g., application programmer interfaces (APIs). A FOSS, as a platform (e.g. a Linux/GNU distribution) provides initially a large set of services and functionality, which is mainly controlled by the distribution designers. Adding more services and functionality is often technically easy at least for technologically savvy users, but may introduce maintenance problems and thus in practice may be available only to a limited user base. In contrast, developing proprietary software for a proprietary operating system (OS) is usually more limited, often only to what the OS provides.

The set of available programming languages is an important characteristic of a platform. Java and scripting languages are very good examples since applications written in them are usually limited to using only services, which are written in those languages (or have a specific interface in the respective language) and services written in another

[4] Note that many proprietary tools can also be extended programmatically.

[5] http://www.ozzie.net/blog/stories/2002/09/24/softwarePlatformDynamics.html

scripting language are typically unavailable[6]. The common division to Java and C/C++ camps is distinct also in FOSS4G but interesting practical solutions exist: Geometry Engine Open Source (GEOS) is a C++ port of the JTS Java Topology Suite (JTS).

From the user's point of view a platform is, roughly, an OS or its user interface (UI), an application, or some specific hardware. A desktop, for example, is a platform where computing mainly happens in one computer, within one OS but usually with more than one application. Application suites, especially so called "office suites" become very popular in the 1980's and 1990's and they are essentially a platform within a platform. The office suites have been more or less proprietary solutions, with only few exceptions, most notably the OpenOffice.org. Spreadsheet tools, as part of an office suite, are a very popular platform for small-scale modeling and management support. Office suites do not have any noteworthy support for working with geospatial data or doing geospatial analysis. A desktop connected to local networks and to the internet is effectively an extended desktop platform. However, if a geospatial system relies on services, e.g., on a spatial database server, which resides in another computer, then it operates on a network or internet platform.

The web is often used as a platform for delivering content such as text, images, and simple applications, intended for wide distribution and use. Some web-based systems are targeting smaller groups and focus more on collaboration. Mobile computing, which is characterized by small and light-weight devices such as cellular phones and personal digital assistants (PDAs), is another platform that is growing in importance and is especially interesting to geospatial computing because of its mobile and location-aware nature.

Each of these software platforms presents an interesting set of benefits and challenges to the user or developer of geospatial software tools and systems. For example, the desktop platform typically has the advantage of allowing for more intensive use of local disk space, memory, and processing power than does a web-based platform. So for large, computationally intensive applications using large data sets, it is generally preferable to deploy geospatial software solutions as desktop based applications. However, web-based applications generally have the advantage of being more rapidly deliverable to end-users and more easily updated. An interesting compromise occurs when a desktop application is developed to be "web-aware" or "web-enabled". In this case, the user gains the benefit of local data processing and storage while using the Internet to download software updates and share data with a wider user community. Such web-enabled desktop tools are growing in popularity (e.g. Google Earth and ESRI's Geography Network).

The Internet has been the focus of much work in the geospatial community. Especially the specifications developed cooperatively in Open Geospatial Consortium (OGC) and the efforts to develop national spatial data infrastructures illustrate this. The FOSS4G community has been active in this work and it has adopted the specifications quickly.

3. FOSS4G SOFTWARE STACK

3.1. Software stack for geospatial computation on the desktop and web platforms

The architecture of FOSS and FOSS4G is usually layered and thus makes up "software stacks". These software stacks can be very deep, the layers at the bottom being, for example, the Linux kernel and the GNU C library, libc. Alternatively, a FOSS4G stack can also have at its base a proprietary product such as Microsoft Windows XP and its associated run-time libraries. Indeed, FOSS4G co-exists with and adjusts to proprietary software easily. Thus it is possible to attract individuals who are forced to use or more comfortable using the Microsoft operating system.

Beside the actual run-time stack, there is also the stack of development tools, like make, which control the compilation of software into binary form, and Glade2, one of many graphical user interface (GUI) development tools. Open source integrated development environments such as SharpDevelop or Eclipse provide support for developers of Microsoft .NET or Java based FOSS tools. The run-time stack consists also of tools like the Apache HTTP server and various libraries. For mixing FOSS and proprietary software on platform level several solutions exist. For example MinGW implements the GNU platform partially in Microsoft Windows. Several FOSS applications are also available for proprietary platforms and/or multiple platforms.

Some software is intended exclusively for the web platform, some is usable on both the web and desktop platforms, and other software is intended primarily for the desktop. Clearly, it is not possi-

[6] The Microsoft .NET Framework and its FOSS counterpart Mono, are notable efforts to overcome this problem.

ble to distinguish absolutely one platform from the other. Rather, we simply attempt to define a typical desktop solution and a typical web solution. The specific and unique software stack that is set up to support geospatial computations makes up the platform together with the hardware and the rest of the computational environment.

Generic Stack	FOSS4G Stack	Grouping
Application Extensions/Plug-ins	Environmental modeling and data analysis tools (e.g. US EPA's BASINS 4, etc.)	End User Application
Application	Quantum GIS, GRASS, OSSIM, JUMP, uDig, MapWindow GIS	User Interface
Application Dev. Environment	Eclipse, QT, OpenGL, SharpDevelop	
High Level Utilities	GeoTools, PostGIS, MapWinGIS ActiveX	Data Serving
High Level Scripting Languages	PHP, Perl, Python, R, R spatial	Data Processing
Low Level Utilities	Shapelib, JTS/GEOS, GDAL/OGR, GMT	
Low Level Languages	C, C++, Java, Fortran, C#, VB.NET	System Software
Operating System	Linux, Darwin, Cygwin, Microsoft Windows	

Generic Stack	FOSS4G Stack	Grouping
Application Web Site	Decision support system, environmental data viewer, etc.	End User Application
Client Side Browser	Firefox, Safari, Netscape	User Interface
Client Side Scripting	JavaScript, Java, Perl, Python	
~~Internet~~~~~~~~~~~~		
Server Side Scripting	PHP, Python, Perl	Data Serving
High Level Utilities	MapServer, GeoServer, GRASS	
Low Level Utilities	Shapelib, JTS/GEOS, GDAL/ OGR, PostGIS, R spatial, GMT	Data Processing
High Level Scripting Languages	PHP, Perl Python	
Low Level Languages	C, C++, Java, Fortran	System Software
Operating System/Drivers	Linux, Darwin, Cygwin	

Figure 1. The tool layers of the desktop platform (above) and the web platform (below) and grouping of the layers. The software list is an example and in some cases is very incomplete. Somewhat similar diagrams have been presented e.g. by Ticheler [2005].

In Figure 1. we present typical software stacks for desktop and web platforms. The system software layers in the two stacks contain many common items. This synergy between the software stacks shows great promise in intertwining capabilities. For example, it is envisioned that future desktop applications will rely more on web-services, while web-based applications will contain more functionality traditionally relegated to the desktop platform. The commonality of the lower layers of the software stacks allows for much of this integration.

We have included Microsoft Windows into the stacks. Although it is not a FOSS operating system, it is the most ubiquitous desktop OS and supports a wide array of FOSS and FOSS4G tools.

3.2 System Software

System software is the foundation on which the complete stack is built, providing software interoperability and a common look and feel for tools. In the web-based FOSS4G software stack, the Linux operating system is the most common OS and has proven to be an exceptionally robust for mapping applications. Linux based geospatial tools for the desktop are also readily available. The Microsoft Windows/.NET framework system software creates an interesting opportunity for FOSS software developers and environmental modelers to build free and open source tools on a common proprietary platform.

System software is typically generic and not directly relevant for geospatial computation. The case of Java vs. other languages is one noteworthy exception. Java is a high-level programming language but it is also often used instead of C or C++ for developing low-level libraries and tools. It is possible to build Java interfaces to C libraries (Java is one of languages supported by Swig, a FOSS generic interface generator) but it is considerably more difficult and not sensible to use Java from other languages. It is also often preferred to have "pure" Java solutions for various reasons. The result has been that software stacks are often divided to Java-based solutions and to other solutions. However, the internet constitutes such a strong dividing line that it is easy to mix Java solutions on the server or client side with other solutions on the other side.

3.3 Data processing

In the data processing layer, data or essential information about it is retrieved into system memory and is processed in some way. The way the data are accessed and what kind of data structures are used is often specific to different FOSS4G tools. One benefit of the FOSS platform is the unobstructed access to these details.

Data processing is divided here into data management, format processing, and geo-analytical processing. Geoprocessing is examined more carefully since it is most important from the point of view of environmental modeling. Data processing tools are quite common between desktop and web platforms.

The foundation of the domain specific interoperability of the geospatial tools is in this layer. Solving of complex problems in EMM requires complex workflows, which usually requires interoperation of several tools in order to be effective.

An example piece of data processing software specific to geospatial computations is a tool to transform datasets from one geospatial coordinate system to another. The most common FOSS4G tool for this is PROJ.4. PROJ.4 contains a system for describing common projections and for converting data between projections.

Data management

Data management is a critical function of GIS for EMM. Both the number of required geospatial datasets and their size are often voluminous. Geospatial datasets are often stored within specialized file formats or data servers. Generally, geospatial data are described and stored in either a vector or raster based model and file format. Raster data are typically stored as a matrix of data, either in tiles or in large singular files in a file system. Vector data are either stored as files or in tables in database management systems. Simple vector data does not consider the topology or connectivity of the spatial primitives and many formats support only this kind of data. Geospatial data sets and information about them, i.e. meta data, are provided for use from servers either through networked file systems or by the http protocol, typically using protocols standardized by the Open Geospatial Consortium (OGC).

Format processing

The GDAL/OGR library and associated utility programs provide widely used basic functionality on the FOSS platform. GDAL provides a generalized API for raster data and a way to implement drivers for raster data formats and OGR does the same for vector data. GDAL can be used to read, access, manipulate, and to write in over 50 raster file formats; OGR enables programs to read, access, manipulate, and to write in over 20 vector file formats and database systems. The GDAL/OGR library is written in C++ with C wrapper and it has been ported to several common operating systems. The OGR library can optionally be compiled to directly include the GEOS library. GEOS gives to OGR some topological capabilities. A Swig interface has been developed and used to create bindings for scripting languages like Python, Perl, and Ruby to the GDAL/OGR library.

Geo-analytical tools

The geo-analytical functionality on the FOSS platform is very strong in theory since FOSS is often the platform of choice for academic research projects. In practice the utilization of all that is possible is sometimes difficult because of interoperability problems and a sometimes steep learning curve to use the tools. Examples of analytical and geo-analytical tools in the FOSS platform include GSL, R, and CGAL. GSL is a general numerical library. The R project develops a language and platform for statistical computing and associated graphics. The R spatial project uses and extends R for spatial statistics. CGAL is a high quality library for computational geometry.

JTS (Java Topology Suite) and its C++ port GEOS is a library for basic computational geometry operations as binary predicates and overlays. Binary predicates return a Boolean value indicating whether two spatial objects have a named spatial relationship. Examples of binary predicates are “intersects” and “within”. Spatial overlays take one or two spatial objects as arguments and return a new spatial object. Examples are “intersection” and “buffer”. GEOS is used by GDAL/OGR and PostGIS (see below). The JTS/GEOS library provides the OGC standard spatial data types. [The JUMP-Project.Org, 2006]

The main analytical method family for raster datasets is map algebra (a term coined by Tomlin [1990]), which extends the standard algebra of scalar values to raster data. For example in map algebra the sum of two similarly projected and sized rasters is a raster, whose cell values contain the sums of respective cell values of the argument rasters. This basic concept may be extended in map algebra into spatial neighborhoods, zones, the 3rd dimension (elevation/depth, voxels), and into raster time series analysis [Karssenberg, 2005]. The GRASS function, “r.mapcalc”, supports the ordinary arithmetic and logical operators, trigonometric etc. functions and it has the concept of cell neighborhood, the r.series command supports time series statistics. libral is a C library, which supports similarly basic map algebra. The Geo::Raster module extends the Perl programming language with map algebra using libral.

More complex geospatial analytical methods include spatial interpolation, terrain analysis (including hydrological analysis), applications of

graph theory (e.g. shortest path computation), spatial data mining (e.g. discovery of spatial phenomena). GRASS is traditionally strong in these areas (see e.g., Neteler and Mitasova [2004]) for example libral and TauDEM provide tools for hydrological analysis.

The most complex geo-analytical tools are those that support spatial modeling. These tools may be implemented with the help of basic geo-analytical methods such as map algebra but they may also work directly on geospatial data. These tools are typically implemented as plug-ins for desktop GIS or as stand-alone applications. Examples include openModeller and SME. openModeller is a spatial distribution modeling library, providing a uniform method for modeling distribution patterns using a variety of modeling algorithms. openModeller can be used via programmatic interfaces, including SOAP and SWIG-python, as well as via a user friendly desktop graphical user interface and as a Quantum GIS plug-in. [openModeller development team, 2006] SME is an integrated environment for developing spatial simulation models [Maxwell et al., 1999].

Visualization is an essential element of geospatial analysis. All GIS provide at least some kind of two- and possibly three-dimensional visualization capabilities, some with animation features. The boundary between mapping and analytical geovisualization is fuzzy. Visualization (mapping) is an important part of data serving and user interface (also analytical geovisualization). Tools for analytical geovisualization are often large standalone applications, like for example the Vis5D or Paraview.

OSSIM is a high performance image processing library and ImageLinker is the GUI application providing access to the OSSIM libraries. ImageLinker is capable of image visualization, of mosaicing a large number of images, and of simple image manipulation tasks like contrast enhancements. ImageLinker lacks some important capabilities, for example image classification and vector capabilities.

Terralib and Terraview are a cross platform Geospatial analysis library and an accompanying front end GUI. Terralib stores all data (including raster) within the RDBMS environment.

3.4 Data Serving

Web servers

The data serving layer exists mainly in the web platform as tools that receive data from the data processing layer and serve it to the user interface layer. The data serving layer is thin but important since it requires standardization and enables a wholly new type of software, i.e., collaborative applications. Collaborative applications are important and will probably become even more important for EMM. Tools such as MapServer and MapGuide, which mainly work in this layer have also become hugely popular in mainstream web-mapping applications.

The two main FOSS tools for serving geospatial data via the web are the MapServer and GeoServer. MapServer can be used to serve maps (as images) and thus create interactive websites, but it can also be used to serve data according to the WMS (map layer) and WFS (vector data) standards. GeoServer supports WMS and WFS-T ("T" denoting transactional, WFS-T supports add, delete, and update of features) specifications.

Spatial databases

In the desktop, this layer is often hidden within an application or does not exist. The only noteworthy exception is the attribute database connection, which often is based on standards (especially SQL).

A relational database management system (RDBMS) is in itself a platform for developing applications, functionality, and services. On the FOSS platform the two main RDBMSs are MySQL and PostgreSQL. A general perception is that MySQL is less featured (in SQL sense) than PostgreSQL but faster. PostgreSQL offers a richer array of DBMS features, such as triggers and it supports a variety of procedural languages. Geospatial data are not easy to store in a standard RDBMS, thus spatial extensions have been developed and standardized by the OGC. PostGIS provides the spatial data types and spatial queries for PostgreSQL, and more recent versions of MySQL have been adding spatial data handling capabilities, too.

Scripting languages

Scripting languages can also be considered as a part of the data serving layer. Scripting languages provide very powerful scripting capabilities besides being general purpose programming languages. Scripting languages can usually be extended by modules, which can be interfaces to data access, graphics or other libraries or they can be extensions written in the scripting language itself. The module system and general purpose

programming capability make scripting languages very useful for "gluing" as an application development methodology. These languages are typically interpreted, and thus programs are easy to write and the language can be used in an interactive fashion. Interactive use of a programming language makes it attractive for "use-developers" [Rosson, 2005]. A use-developer is anybody, who creates applications with spreadsheets, interactive web-pages, or, as in this case, with a few lines using a scripting language.

Many scripting, interpreted programming languages, such as Perl, PHP, Python, Ruby, Tcl, and R have been developed as FOSS and thus the FOSS platform is well suited for their use. The R language is developed for statistical computing but it is also a general purpose programming language. These languages also support modular and object-oriented programming for creating larger applications. The openness of these languages has attracted developers to create and publish extensions to them, thus making, e.g., database integration, data import, and development of graphical applications relatively easy. The main problem is that the tools developed for Python for example, are not directly usable by people who use for example Perl.

3.5 User Interface

The user interface is often, but not necessarily, significantly different in the desktop and in the web platform, the web platform being dominated by the web browser with its own set of menus, toolbar buttons and intrinsic functionalities. Some tools (e.g. uDig), because of their platform independent nature or their significant client server interaction have begun to blur the distinction between a desktop and web end-user application.

Lately two modern desktop GUI GIS that are FOSS have appeared: Quantum GIS and MapWindow GIS. Both have a standard menu system, graphical dialogs, and other desktop software GUI elements common to proprietary GIS software. Both have a plug-in architecture for extension developers, making them suitable for supporting environmental models and modeling toolkits.

MapWindow GIS is the first fully Microsoft .NET compatible open source GIS and has the advantage of providing developers with ActiveX and .NET components that can be used in many common programming languages (i.e. Visual Basic, C#, Delphi, VBA, etc.) The current MapWindow GIS development effort is centered on producing an enhanced suite of geoprocessing, data access and visualization components that can be used in both desktop and web-based (using ASP.NET) applications.

Quantum GIS has the notable advantage of being fully cross-platform such that the same tools run on Microsoft, Macintosh, and Linux/GNU operating systems. An effort to make Quantum GIS usable as a GUI for GRASS is making it possible to exploit GRASS' analytical capabilities directly from Quantum GIS.

Mapnik and Gtk2::Ex::Geo are FOSS toolkits for developing geospatially aware applications. Mapnik focuses on cartographic quality and Gtk2::Ex::Geo focuses on providing a geovisualization widget and combining GUI and CLI. Mapnik builds on a graphics library called AGG, which is FOSS. Gtk2::Ex::Geo is a collection of Perl modules and it builds on Gtk2-Perl software and Geo::Raster and Geo::Vector modules, which in turn build on libral and GDAL/OGR.

OpenEV is a general viewer for geospatial data, and provides for example basic vector overlay capabilities. OpenEV's analytical capabilities are limited and for the most part they have to be accessed via the command line.

SAGA GIS is a desktop GUI environment and library for geoscientific analysis.

uDig represents a new approach to building a desktop GIS as it treats web-based (OGC compliant) data sets as similarly as possible to local data sets. uDig is also designed so that new, derived applications can be created by developers.

3.6 End-user applications

At the top of the FOSS software stack are end-user applications – the tools that are used by managers, modelers, stakeholders and others to better understand the environment. These can be as simple as a customized map viewer on a web site showing the location of a particular environmental problem, or management area, or as complex as a fully integrated dynamic simulation watershed model. In each case, it is at this top layer where end users interface with the software. Hence, this layer needs to be a critical focal point for the FOSS4G community, so that the tools developed at the underlying layers meet the actual (not only perceived) needs of end users.

Interestingly many existing tools in this layer are already (and in many cases have been for years) FOSS tools. For example, SWAT, HSPF, WASP, QUAL2E and other notable hydrologic simulation

models have always been FOSS. However, due to the lack of suitable FOSS4G tools to support these models (and because many were developed prior to the advent of GIS) current implementations of these models on FOSS4G platforms is limited. In other words, it is possible to see the source code to the model, but not the to the proprietary GIS platform upon which the model can run. This problem is being addressed in many cases as efforts are underway to generate FOSS4G interfaces for these models.

4. WORKFLOWS FOR ENVIRONMENTAL MODELING AND MANAGEMENT

4.1 Introduction

Environmental modeling and management (EMM) are both activities, which increasingly happen computationally with desktop and web-based tools. In this chapter we examine the workflow concept and user classes, we define a few representative EMM workflows that involve mapping or geospatial analysis, and we analyze how the FOSS tools are suited to the task. In the cases we also discuss some ongoing efforts to link FOSS4G and EMM.

The goal in environmental modeling is to develop and use a model (for our purposes, we use the term "model" to mean a computer based generalization of a real system) of an environmental system. Important domains in environmental modeling include geospatial, temporal, geophysical, chemical, and ecological. How geospatial domains can and should be included in modeling has to be assessed by the modeler. The historical limited availability of spatial data has caused severe limitations to modeling; the situation is getting better thanks to new remote sensing instruments and the development of standard datasets but problems remain due to reduced budgets and copyright restrictions.

The motivation for modeling environmental systems may be for example scientific interest, impact assessment, planning, or environmental management. A model is in practice either a descriptive model, a simulation model, or an optimization model. A descriptive model is for example a database, which has a schema and which can be queried. A simulation model explains or predicts the behavior of a system over time (or space). An optimization model can be used to find the best values for decision variables.

Environmental management typically has two phases: learning and deciding. Learning may mean simple assimilation of data or it may mean development of assessment skills. Descriptive and simulation models are often used for learning. Decision making is the task of selecting between alternatives, but the more time consuming tasks of inventing the alternatives and the assessment of their impacts are usually included in decision making. Simulation and optimization models can be used to invent or refine alternatives. Simulation models are routinely used for impact assessment. Optimization models can be used to suggest decisions once evaluation criteria and methods for computing them are selected.

The requirements that environmental modeling and management set for geospatial tools and methods may be organized according to the task at hand:

- technical tasks: storage of data, format conversion, etc,
- supporting simple assimilation of data: view, visual overlay, etc,
- a formal language: writing of specifications, programming a model, etc,
- planning support: sketching of alternative spatial plans,
- analytical tasks: preparation of input data, execution of model, evaluation of model output,
- support for assessment: expert advice, decision support, probabilistic reasoning, evaluation of plans.

Other requirements are very diverse and stem from the type of the project, its goals, and work habits:

- requirements on the user interface: what are the computer skills of the user of the tool
- computation time: will the tools be used in interactive sessions for example,
- support for cooperation: will one user do everything or will there be several users with different skills.

4.2 User Classes

End users of software typically have the expectation that there is a GUI which resembles in behavior and appearance (as closely as possible) GUIs of the other software that they use. The need to study help texts and manuals is typically seen as a hindrance. One classification for users is the Rosson's [2005] developer, user-developer, and user.

Another classification is by the time the user can devote to using a tool: 10 minutes, one hour, one day, etc. There is no universal classification of users, the issue has to be considered in the beginning of each tool development, modeling, and management project.

Nielsen [1993] classifies existing user interface types and associated interaction styles. The interaction styles he lists are no interaction, question-answer, command language, function keys, form fill-in, menus, and direct manipulation (with a mouse for example). The ease of use of a GUI comes from the fact that there is only a limited set of possible actions and they are presented to the user. Command line interface (CLI) provides added functionality mainly because of the much larger set of possible actions and the possibility to combine tasks. CLI is possible to combine with GUI using free-form text input. CLI also leads to scripting which is essentially the stored (and documented) sequence of individual commands with optional use of variables for task generalization. Coupling these methods enables the information system to support sophisticated data processing workflows.

Developers are typically looking for a clean set of API's to which they can integrate custom programming interfaces.

The current set of FOSS4G tools represents a broad spectrum of support for various users. Fortunately, there is a clear trend in the community to develop tools that are easier to use and more similar to other common software tools with respect to user interaction (e.g. they support standard mouse behavior and key sequences, and have common types of "GIS oriented" tool bar buttons for map navigation, etc.)

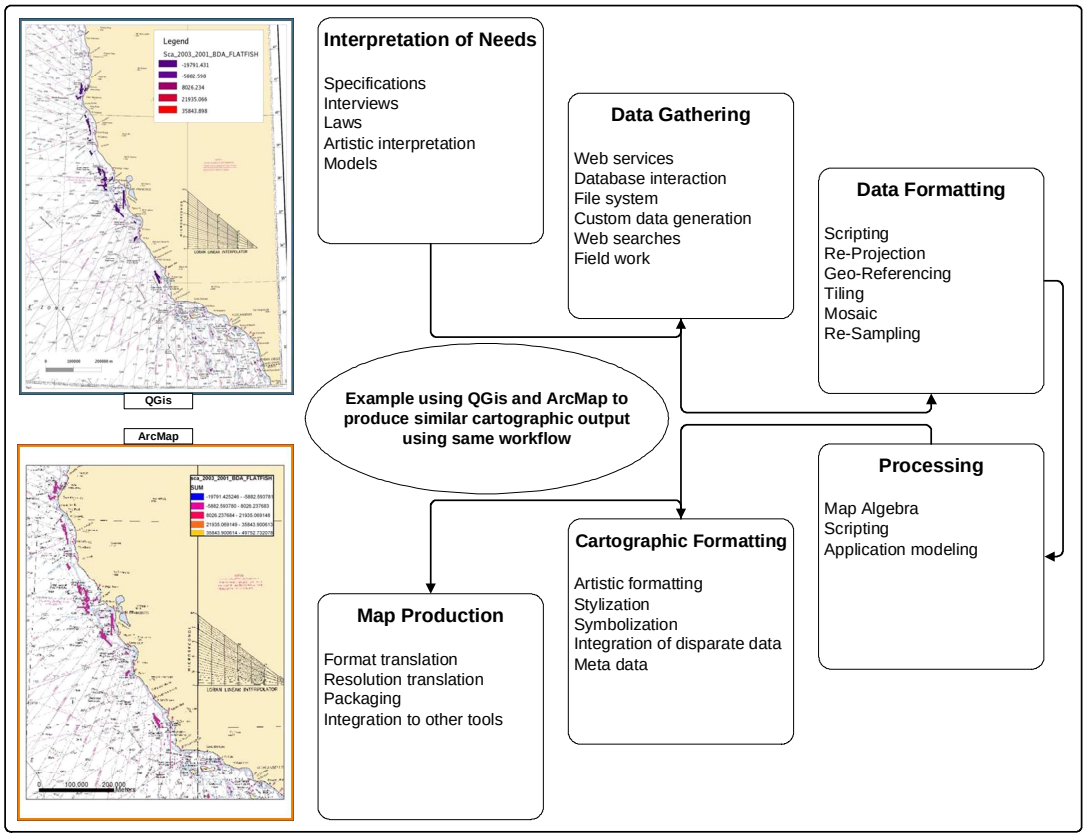

Figure 2. Example of the cartographic workflow producing map output in both FOSS and proprietary software. Data is depicting change in fishing effort by the Flatfish fishery off the California coast between 2001 and 2003, during which time a fishing closure was instituted along the continental shelf.

As the environmental modeling community begins to adapt these tools to their specific modeling and end user needs, it is expected that the interfaces will continue to improve and be refined such that they are more acceptable to a wider array of users from all user classes.

4.3 Case 1 – Cartographic map production

The workflow to produce a map is initiated by an expressed need. In our case the need is related to EMM. In this case we define a map as a static visualization, essentially an image file, which typically combines geospatial information from various sources. Map production has traditionally been one of the main functions of a GIS. The workflow consists of the following steps:

1. Interpretation of the expressed need for a map.
2. Gathering of required data.
3. Formatting of gathered datasets.
4. Processing of datasets.
5. Cartographic formatting of the requested map.
6. Production of the requested map.

Maps are arguably more useful in management than modeling, although graphical visualization of all geo-spatial processing is useful for data validation and understanding. Maps are useful both in learning about the case or system at hand, and at supporting decision making.

In general the FOSS4G tools are good at steps 2 to 4 but support for steps 5 and 6 are more limited. (Figure 2.) With tools like Mapnik and GIMP there is a good chance that these steps will be better supported in the future. Also, we expect that many traditional mapping applications resulting in paper maps will be replaced by interactive web based mapping in the future.

4.4 Case 2 – Web-based mapping

Web-based mapping is effectively a simplified, or canned, form of cartographic map production. The creation of maps can be done within a framework offering a limited number of datasets and layout options. Because of these limitations the process of creating a map is more straightforward than traditional cartographic map production (Figure 3.).

Reasons for using web-based mapping to solve problems in environmental modeling and management include:

1. Instant access to the latest version of analytical analysis and models by geographically separated teams.
2. Collaboration in management can be achieved over the web.
3. Interactiveness that can be built into the web-mapping site.
4. Cross platform nature and ease-of-use of browser-based solutions.

The FOSS4G solutions for web-based mapping are very good mainly due to the popularity of MapServer and MapGuide Open Source. Also notable as free though not open source, is the Google Map API which currently allows for free hosting of simple maps (appropriate for basic navigational needs) on any publicly accessible website.

The analytical capabilities of web-based mapping solutions on the FOSS platform are limited by technical problems. Solutions which use GRASS as a backend for MapServer exist but are not optimal in a technical sense. The development of scripting language interfaces to geo-analytical libraries and the development of the actual analytical libraries both in C and in Java have a potential to solve this problem. The former will enable server-side solutions and the latter (Java) will also enable client-side solutions.

4.4 Case 3 – Numerical Simulation

Environmental simulation models often require pre or post-processing of geospatial data, or they can be tightly linked to a GIS, using it as a GUI. Harvey and Han [2002] have presented an excellent analysis of the relevance of FOSS to hydraulic and hydrological models. Several environmental models are available as FOSS, for example MODFLOW, which is a groundwater simulation model, and SWAT, which is a river basin scale land management impact assessment model. It is interesting to note that although SWAT is FOSS, it uses a proprietary GIS for a GUI (although this is currently changing – see below). A general impression (supported by discussions with modelers) is that FOSS models are popular and get used.

The general workflow of modeling is

1. Setting of objectives and scope and scale of the modeled system.

2. Data collection.
3. Development or selection of a model.
4. Preparation of input data.
5. Calibration and validation of the model.
6. Using the model.
7. Analysis of the model output.

Geospatial tools are typically needed at steps 2, 4, 7. (Figure 4.)

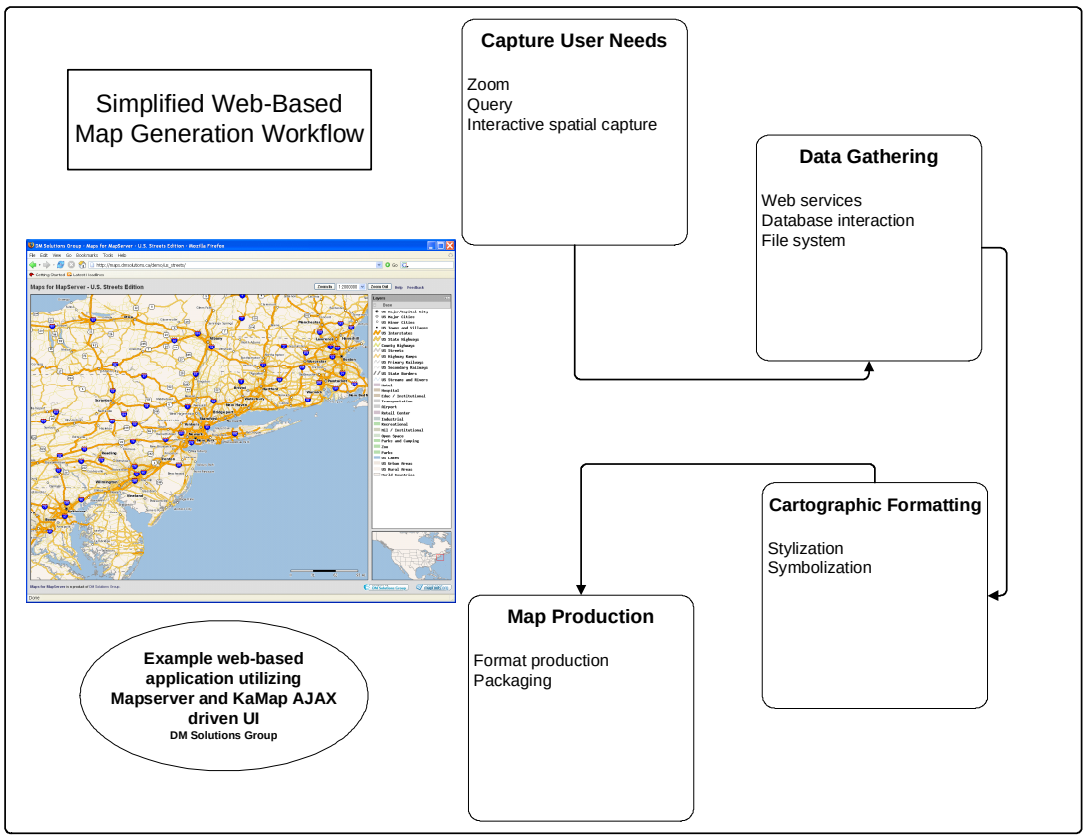

Figure 3. Example of web-based geo-spatial interface and workflow.

Three interesting and complementary efforts to merge environmental modeling with FOSS4G tools are currently under way at the United States Environmental Protection Agency (US EPA) Office of Science and Technology (OST), US EPA Office of Research and Development (ORD), and the United States Department of Agriculture (USDA).

In each case an existing environmental model or modeling system is being adapted to explicitly use geospatial data in a FOSS4G based GIS environment. Specifically, OST has recently awarded a five year contract to a consortia of consulting firms and universities to improve and provide support for the BASINS watershed modeling system. The awarded team specifically proposed migrating BASINS from a proprietary GIS system to a FOSS4G GIS system (namely MapWindow GIS). Similarly ORD is currently investing in the adaptation of MapWindow, GDAL, JTS, and other FOSS4G tools to support a wide variety of EPA environmental modeling tools, beginning with the FRAMES/3MRA system. USDA through it's Texas A&M university collaborators are following suit with the development of a new GIS interface to its SWAT watershed model, again based on MapWindow and other FOSS4G tools.

4.5 Case 4 – Environmental management

Environmental management workflows consist of:

1. Monitoring the state of the environment.
2. Planning of actions for improving the state.

3. Responding to actions, which affect the environment.

4. Increasing awareness of people of the state of the environment

The workflow of environmental management can be split into the above parts, all of which consist of independent workflows described in the first three cases of this section. This long-term workflow is perhaps best supported with developing a comprehensive geospatial database of the system, which is managed. PostGIS is one example of a tool that is well suited for this purpose. It is possible to build various stacks on the top of PostGIS to support real-time or ongoing monitoring, analytical needs, decision making, and mapping for delivering information.

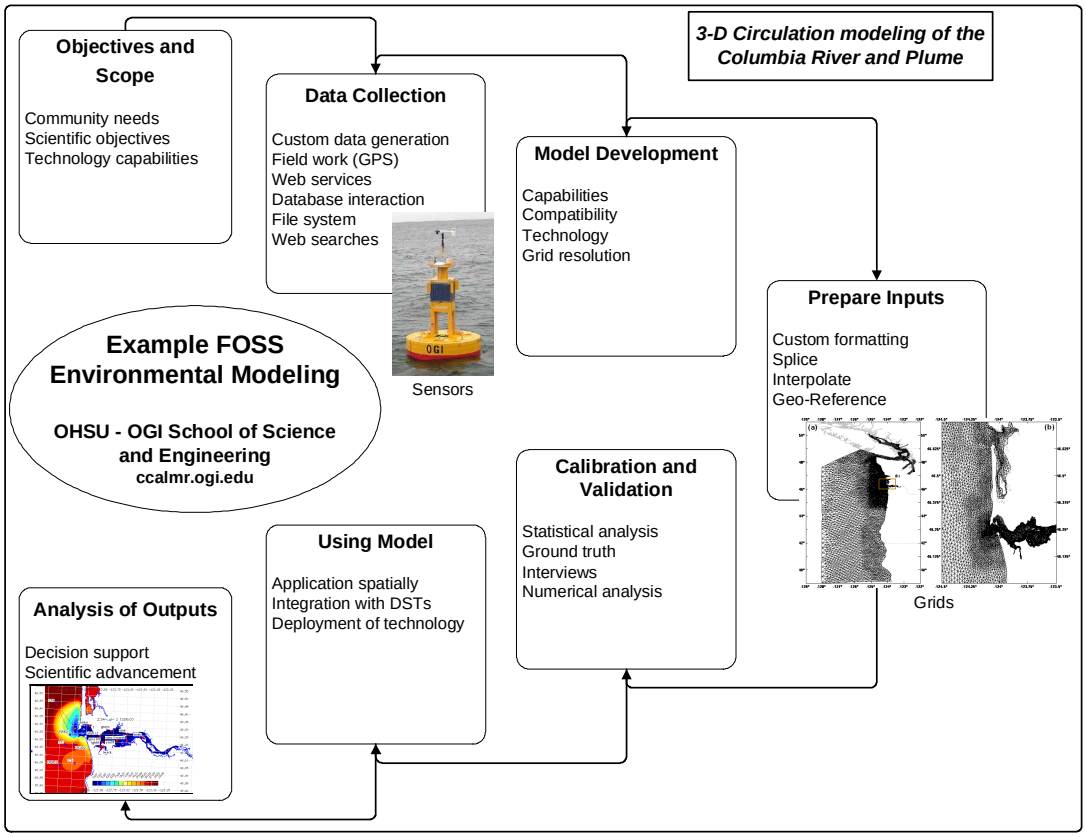

Figure 4. Example FOSS based modeling system integrating spatial data and GIS tools such as PostGIS, OGR/GDAL, and Mapserver. 3-D circulation models are used to simulate and forecast physical parameters that are then used for environmental and ecosystem management.

In the domain of environmental management most projects and project proposals currently include a component, which aims to develop software and databases. These components may be quite large in large international cooperative projects. For example there is an International Groundwater Resources Assessment Centre (IGRAC), which works under auspices of UNESCO and WMO. Perhaps the most important part of the work of IGRAC is the development of Global Groundwater Information System (GGIS). This information system draws on several other initiatives and data collection efforts. Many projects similar to IGRAC's GGIS exist. The potential amount of data in such databases and accessible by such information systems may be huge. Also the complexity of the data may be overwhelming calling for dedicated meta data projects.

Many international organizations and initiatives such as FAO [Ticheler, 2005] are currently investigating FOSS4G for their needs in developing Internet-based information systems. Another example of a similar effort is the Managing Information for Local Environment in Sri Lanka (MILES)

project[7]. MILES has run a virtual seminar on issues concerning the linking of FOSS4G and environmental management.

5. DISCUSSION

As computing becomes more ubiquitous, the significance of one single tool becomes less important and focus shifts more towards software stacks and platforms. It is difficult to assess how much of this is conscious "empire building" by some software companies and how much is natural evolution. Organizations are clearly making strategic decisions on software platforms and stacks that they use and support. Although it is a believable trend that more users and in fact user-developers, the platform on which they develop is often very restricted and dictated by institutional guidelines.

The dividing line between FOSS and proprietary software is fuzzy, partly because it is in the interest of developers of proprietary software to make it fuzzy and partly because the end-users are getting more and more reluctant to buy software. People are expecting web browsers and other common tools to be free of charge. Also, depending on license of the particular FOSS tool, proprietary tools may include FOSS. Advances in standards aiming for interoperability make it possible to co-use free and proprietary tools.

It is easy for a tool developer to fall into the trap of slowly expanding the scope of his tool, promoting the solving of ever bigger and more complex problem with his tool. From the user's point of view the benefit in this evolution is that he can "grow with the software" not having to go through the learning curve of other tools. The developer of a proprietary product is often happy with this, since he or she does not have to share income with others. After a long experience in this kind of familiar worlds, the FOSS world must look chaotic and complex.

In the FOSS world the barriers to interoperability are much lower and thus the software stack tends to be thicker in FOSS platform than in the proprietary platform. There is competition in the FOSS4G world, but it is not preventing evolution of individual tools, stacks, or platforms. Code sharing is encouraged, as exemplified by activities within so called "foundation projects" in the OSGeo Foundation. The competition in the FOSS4G world seems to happen on two distinct domains: on community development and on technical and usability merit.

Our examples show that it is possible to build software stacks from current FOSS4G to support EMM workflows. IT professionals have long embraced the concepts of platforms and software stacks, now the same is happening in the domain of EMM. It is important to include these concepts into the communication between EMM and FOSS4G communities. This communication for example conveys the needs and requirements of the EMM practitioners to software developers, who can analyze them and understand what needs to be done. Common platforms and interoperability are important for the users. When design is based on selection among proprietary products, the software stack concept is usually not usable. The result is that decisions may be based on somewhat questionable analysis, for example based on just supported platforms and brand names etc.

FOSS4G still caters in many cases to advanced users. The FOSS4G solutions for application areas are usually not as "packaged" as those offered for proprietary products, although companies exist, which specialize on delivering complete FOSS solutions. This is partly a fundamental difference but it may also change if and when people working in application areas discover FOSS4G. Additionally it is incumbent upon developers of FOSS4G tools to improve the ease of use, installation, and integration of such tools so that they can be more readily adapted by the environmental modeling community. Potential improvements might include:

- Providing compiled binaries for multiple platforms and operating systems,
- Developing demonstration applications that show how one might integrate FOSS4G tools with legacy FORTRAN or other existing environmental modeling code,
- Generating simplified installation packages that can be readily adapted and integrated with the installation package of a particular model,
- Enhancing existing user communities and developing new discussion forums targeted specifically at FOSS4G users in the environmental modeling community,
- Clarifying the meaning and interpretation of various FOSS license agreements, and

[7] http://www.miles.geo.ar.tum.de

- Seeking out opportunities to adapt FOSS4G tools to the specific needs of the EMM community.

In the examples we mention above in Case 3, environmental numerical modeling, the particular funding agency has chosen FOSS4G solutions because of the opportunities to redistribute resulting modeling tools freely to end-users and to support general goals of openness and transparency with respect to modeling tools. Indeed, the main marketing advantages of FOSS4G are its low cost, open licensing, and modular nature allowing diverse tools.

Certainly, FOSS4G has been around and evolved for a very long time, GRASS is the prime example of that. The current challenge for FOSS4G is to develop working software stacks, which provide solutions that are attractive to end-users and people working in the application areas. The open data formats and data exchange protocols are currently shaping the industry and FOSS4G are proving to be very good with them.

6. CONCLUSIONS

FOSS4G has many advantages, which should be attractive to modelers and managers like low cost, easy dissemination, code transparency, natural support for extensions and experimentation. The greatest barriers for their increased use in the environmental modeling and management community seem to be the (sometimes) perceived (low) importance of geospatial aspect, some technical obstacles, and low visibility.

In this paper we have identified and described elements in environmental modeling and management, which require or benefit from geospatial computation. We have also described geospatial software, focusing on FOSS tools and solutions, and we have discussed how these tools can be used to accomplish tasks of environmental modeling and management.

7. ACKNOWLEDGEMENTS

The authors would especially like to thank the people at the various email lists: e.g., Geowanking, GRASS users, OSGeo discuss, OpenNR, freegis list, etc. for their ideas, visions, and enthusiasm. Many of these people have created weblogs, wikis, and websites which have provided invaluable background information for this paper. Some of the ideas presented in this paper no doubt originate in some of these sources. The Open Geospatial Consortium should also be acknowledged for its work in developing and publishing open specifications, which has been a great motivation for many FOSS4G developers.

8. REFERENCES

Anonymous, The free software definition. http://www.fsf.org/licensing/essays/free-sw.html (URL accessed 26.4.2006)

Bonnici, K., To international markets with competitive projects. *Avopaikka, Open source business news* 1/2006. Finnish Centre for Open Source Software. On page 6.

Daratech Inc., Press Release: Worldwide GIS Revenue Forecast to Top $2.02 Billion in 2004, up 9.7% over 2003, http://www.daratech.com/press/releases/2004/041019.html (URL accessed 25.4.2006)

Gross, T., Hood, R., Voinov, A., Building a Community Modeling Culture. (in prep.)

Harvey, H. and Han, D., The relevance of Open Source to hydroinformatics. *Journal of Hydroinformatics*. (4) 2002. pp 219-234. also available at http://public.hamishharvey.fastmail.fm/publications/2002-10-jhydroinf-open-source/open-source-hydroinformatics.pdf (URL accessed 10.3.2006)

Karssenberg, D. and De Jong, K., Dynamic environmental modelling in GIS: 1. Modelling in three spatial dimensions. *Int. J. Geog. Inf. Sci.* 559-579. 5(19) 2005.

Maxwell, T., Villa, F., and Costanza, R., Spatial Modeling Environment. http://www.uvm.edu/giee/SME3/ (URL accessed 26.4.2006). The SME software is available at SourceForge: http://sourceforge.net/projects/smodenv (URL accessed 26.4.2006).

Neteler, M. and Mitasova, H., Open Source GIS: A GRASS GIS Approach. Second Edition. The Kluwer international series in Engineering and Computer Science (SECS): Volume 773. Kluwer Academic Publishers / Springer, 420 pp. Boston, 2004.

Nielsen, J., Usability engineering. Morgan Kaufmann. 362 pp. 1993.

Rosson, M.B., The end of users. Keynote presentation at OOPSLA 2005 conference, October 16-20, 2005, San Diego, California.

Ramsey, P., The State of Open Source GIS, http://www.refractions.net/white_papers/index.php?file=2005-2-2_oss_briefing.data (accessed 24.4.2006)

The JUMP-Project.Org, Project overview. http://www.jump-project.org/project.php?PID=JTS&SID=OVER (URL accessed 10.3.2006)

Ticheler, J., SDI Architecture diagram. http://193.43.36.138/relatedstuff/index_html/document_view (URL accessed 10.3.2006)

Tomlin, C.D., Geographic Information Systems and Cartographic Modelling. Prentice-Hall. Englewood Cliffs, NJ, 1990.

Wikstrøm, M. and Tveite, H. 2005. PostgreSQL/PostGIS and MapServer compared to ArcSDE and ArcIMS in performance on large geographical data sets. *Kart og plan* (3) 2005. pp 185-192. (in norvegian)