The State of Open Source GIS

Prepared By: Paul Ramsey, Director
Refractions Research Inc.
Suite 300 – 1207 Douglas Street
Victoria, BC, V8W-2E7
pramsey@refractions.net
Phone: (250) 383-3022
Fax: (250) 383-2140

Last Revised: September 15, 2007

TABLE OF CONTENTS

1 SUMMARY

1.1 Open Source

"Open source" software is technically defined as software in which the source code is available for modification and redistribution by the general public. There are a myriad of different open source software licenses, and the "Open Source Initiative" (http://www.opensource.org/) has taken on the role of general arbiter of license correctness.

It is easy to become overly distracted by licenses and source code when evaluating open source software (OSS), or considering OSS as a corporate or project strategy. Fundamentally, successful OSS projects are not created by releasing free source code – they are created through the growth of *communities of shared interest*.

For example, Apache is not a successful open source project because the code is freely available. There are numerous web server projects that have freely available and open source code. Apache is the preeminent open source web server because it commands a powerful community that shares an interest in maintaining Apache as a top-drawer web server. The Apache community includes corporate giants like IBM and HP, government agencies, and academic contributors. It also has a role for individual contributors. These diverse actors can work together collaboratively because the Apache software and the Apache organization have been engineered together to maximize transparency and openness:

- **The software itself is designed in a modular manner**. At a basic level, contributors can aid the project by writing special purpose modules which add otherwise obscure functionality. For example, mod_auth_pgsql allows Apache to do basic HTTP authentication by reading user names and passwords from a PostgreSQL database. This is obscure functionality, usable by maybe a few thousand users, but it adds an incremental value to the product, and the modularity of the software makes it easy to add.
- **The software is extremely well documented**. A successful project must reduce the amount of friction experienced by new contributors to a minimum, to maximize the amount of useful effort directed at the project. Time spent figuring out undocumented software internals is time not spent productively working on the code.
- **The software core design and development process is transparent**. All the mailing lists used by the core team for discussions of design ideas and future directions are public. Anyone can contribute to the discussion, although the core team will make the design decisions in the end. The source code is

available throughout the development process, via a CVS (concurrent versioning system) archive, not just at release time.

- **The core team itself is modular and transparent**. The core development team is made up of programmers who self-select. New members are added based on their contributions to the source code. When a core member ceases contributing to the project, they are removed after a set time period. There is a governance structure that openly allows access to the core team based on programming merit, not corporate or government affiliation.

The strength of open source projects therefore should be evaluated not simply on technical merit or on legal license wording. OSS products should be evaluated like COTS ("commercial off-the-shelf") products, comparing both the technical features and the vitality of the community that maintains and improves the project.

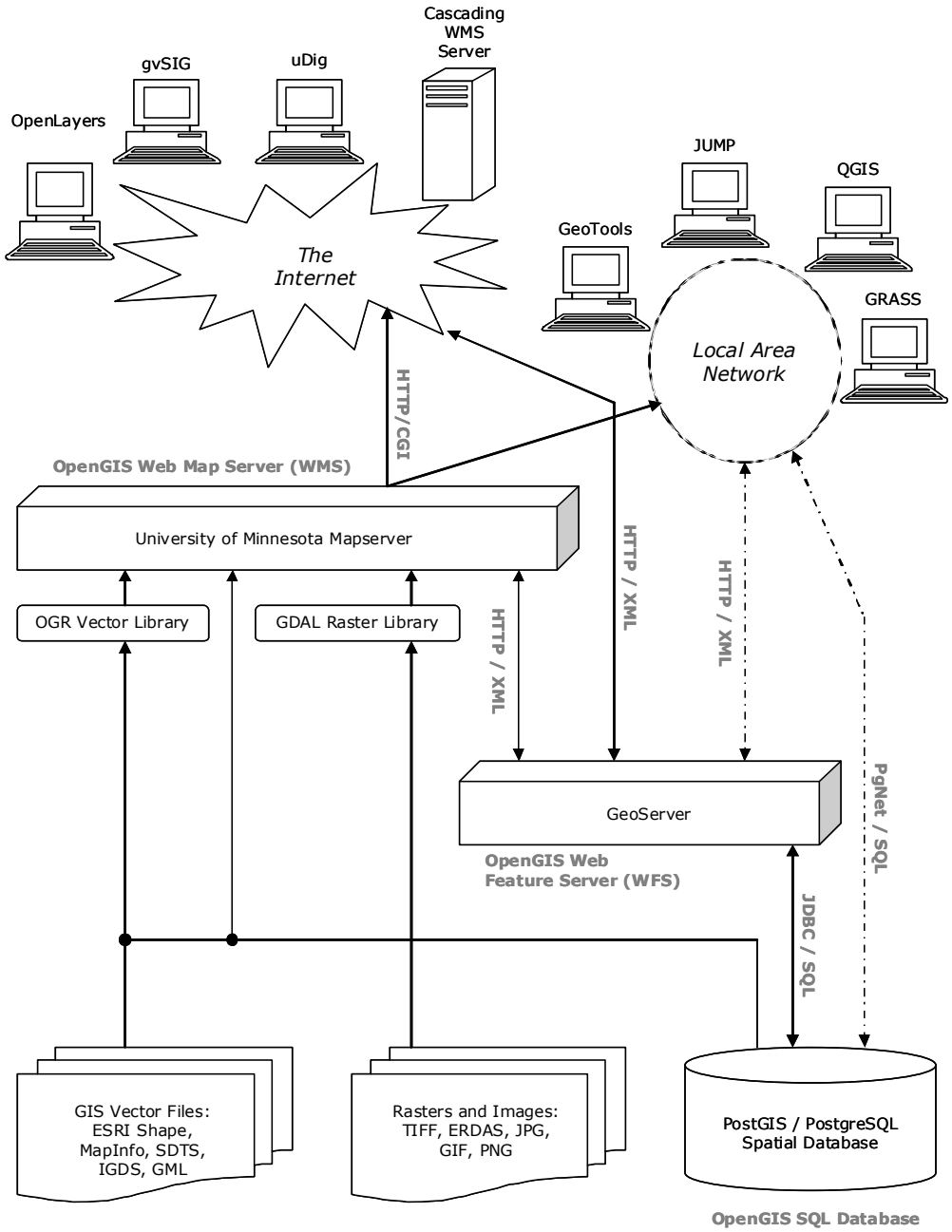
Evaluations of OSS projects should ask:

- **Is the project well documented?** Does the web presence provide direct access to both the source code and documentation about the internals of the code? Is there tutorial level documentation for all three user categories (user, administrator, programmer) to get people up and working with the software quickly?
- **Is the development team transparent?** Is it clear who the core development team is? Is the development team mailing list public? Is the current development version of the code available online? Is membership in the team attainable via a merit-based process?
- **Is the software modular?** (This criterion is more applicable to some projects than others, depending on design constraints.) Is there a clear method to add functionality to the project that does not involve re-working the internals? Is this method documented clearly with examples? Is there a library of already-contributed enhancements maintained by the wider user / developer community?
- **How wide is the development community?** Are multiple organizations represented in the core development team? Are core team members financially supported in their work by sponsoring organizations? Is the development community national or international? How large is the user mailing list? How large is the developer mailing list?
- **How wide is the user community?** (This criterion is basically a standard COTS criterion – more installations imply wider acceptance and testing.) What organizations have deployed the software? What experiences have they had?

The more of these questions which are answered in the positive, the healthier the OSS project under examination is.

1.2 Open Source GIS

The Open Source GIS space includes products to fill every level of the OpenGIS spatial data infrastructure stack. Existing products are now entering a phase of rapid refinement and enhancement, using the core software structures that are already in place. Open Source software can provide a feature-complete alternative to proprietary software in most system designs.



2 IMPLEMENTATION LANGUAGES

Open Source GIS software can be categorized into a few largely independent development tribes. Within each tribe, developers cross-pollinate very heavily, contribute to multiple projects, and have high awareness of ongoing developments. The tribes can be loosely described as:

- The 'C' tribe, consisting of developers working on UMN Mapserver, GRASS, GDAL/OGR, OSSIM, Proj4, GEOS, PostGIS, QGIS and MapGuide OS. The 'C' tribe also includes users of scripting languages that bind easily to C libraries, such as Python, Perl and PHP.
- The 'Java' tribe, consisting of developers working on GeoTools, uDig, GeoServer, JTS, JUMP, and DeeGree.
- The '.Net' tribe, consisting of developers working on Worldwind, SharpMap, NTS, and MapWindow.

The PostGIS/PostgreSQL project – by virtue of standard database interfaces like libpq (C/C++), ODBC, NPgSQL (.Net) and JDBC (Java) – is used by the tribes more or less equally. However, because it is written in C, PostGIS is a natural member of the C tribe and uses many of the C-based GIS support libraries. Mapserver is used by some Java developments via JNI (Java Native Interface) bindings, or via the OpenGIS WMS and WFS protocols.

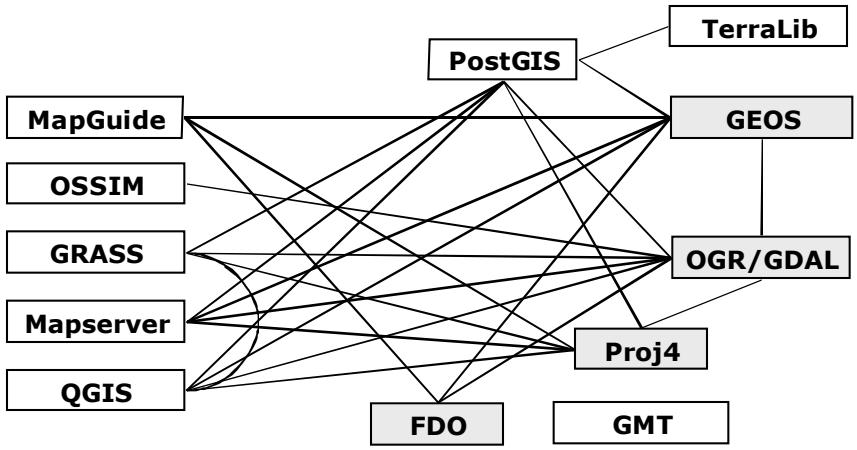
Both the C and Java development areas have a high degree of internal project linkage, with a great deal of leverage being applied through code reuse and linking libraries.

The .Net area has the advantage of re-using code from both the Java and C areas, through the mechanism of .Net "assemblies" (to wrap C/C++ code) and cross-compilation of Java code using J#.

Finally, there is a wild card category of projects that do not fall into language bins: web applications. This category includes various toolkits and web services that provide a browser-based interface to spatial web services, like mapping servers.

2.1 Survey of 'C' Projects

The 'C' projects are, in general, more mature than the Java and .Net projects, having been in development for a longer period of time, and having had more time to attract active development communities. The core of the 'C' projects are the shared libraries (shown in grey below), which are re-used across the application space and form the base infrastructure for common capabilities, such as format support and coordinate re-projection.



2.1.1 Shared Libraries

The shared libraries provide common capabilities across the various C-based applications, allowing applications to easily add features that would ordinarily involve a great deal of implementation.

2.1.1.1 GDAL/OGR

The GDAL/OGR libraries are really two logically separate pieces of code: GDAL provides an abstraction library for raster data and modules for reading and writing various raster formats; OGR provides an abstraction library for vector data and modules for reading and writing vector formats. However, the two libraries are maintained within the same build system for historical reasons and because both libraries are maintained by the same person.

Maintainer: Frank Warmerdam (warmerdam@pobox.com)

Web Site: http://www.gdal.org/

Implementation Language: C++

Source License: MIT

Because the source license for GDAL/ORG is BSD, the library is also used in several proprietary GIS packages, and the maintainer derives some income through maintaining the capabilities of the package for these proprietary users.

GDAL supports the following raster formats and many others:

Long Format Name	Code	Creation	Georeferencing	Maximum File Size
Arc/Info ASCII Grid	AAIGrid	Yes	Yes	No limits
Arc/Info Binary Grid (.adf)	AIG	No	Yes	--
Microsoft Windows Device Independent Bitmap (.bmp)	BMP	Yes	Yes	4GiB
BSB Nautical Chart Format (.kap)	BSB	No	Yes	--
CEOS (Spot for instance)	CEOS	No	No	--
DODS / OPeNDAP	DODS	No	Yes	--
Military Elevation Data (.dt0, .dt1)	DTED	No	Yes	--
ERMapper Compressed Wavelets (.ecw)	ECW	Yes	Yes	
ESRI .hdr Labelled	EHdr	No	Yes	--
ENVI .hdr Labelled Raster	ENVI	Yes	Yes	No limits
Envisat Image Product (.n1)	Envisat	No	No	--
EOSAT FAST Format	FAST	No	Yes	--
FITS (.fits)	FITS	Yes	No	
Graphics Interchange Format (.gif)	GIF	Yes	No	

Long Format Name	Code	Creation	Georeferencing	Maximum File Size
GRASS Rasters	GRASS	No	Yes	--
TIFF / GeoTIFF (.tif)	GTiff	Yes	Yes	4GiB
Hierarchical Data Format Release 5 (HDF5)	HDF4	Yes	Yes	2GiB
Erdas Imagine (.img)	HFA	Yes	Yes	No limits
Atlantis MFF2e	HKV	Yes	Yes	No limits
Japanese DEM (.mem)	JDEM	No	Yes	--
JPEG JFIF (.jpg)	JPEG	Yes	Yes	4GiB (max dimentions 65500x65500)
JPEG2000 (.jp2, .j2k)	JPEG2000	Yes	Yes	
NOAA Polar Orbiter Level 1b Data Set (AVHRR)	L1B	No	Yes	--
Erdas 7.x .LAN and .GIS	LAN	No	Yes	2GB
Atlantis MFF	MFF	Yes	Yes	No limits
Multi-resolution Seamless Image Database	MrSID	No	Yes	--
NITF	NITF	Yes	Yes	
NetCDF	NetCDF	No	Yes	2GB
OGDI Bridge	OGDI	No	Yes	--
PCI .aux Labelled	PAux	Yes	No	No limits
Portable Network Graphics (.png)	PNG	Yes	No	
Netpbm (.ppm,.pgm)	PNM	Yes	No	No limits
USGS SDTS DEM (*CATD.DDF)	SDTS	No	Yes	--
SAR CEOS	SAR_CEOS	No	Yes	--
USGS ASCII DEM (.dem)	USGSDEM	No	Yes	--
X11 Pixmap (.xpm)	XPM	Yes	No	

OGR supports the following vector formats:

Format Name	Creation	Georeferencing
Arc/Info Binary Coverage	No	Yes
CSV (Common Separated)	Yes	No
DODS / OPeNDAP	No	Yes
DWG/DXF	Yes	No
ESRI Shapefile	Yes	Yes
ESRI ArcSDE	No	Yes
ESRI Geodatabase	No	Yes
GML	Yes	No
GRASS	No	Yes
Mapinfo File	Yes	Yes
Microstation DGN	No	No
MySQL	Yes	Yes
ODBC	No	Yes
Oracle Spatial	Yes	Yes
PostgreSQL	Yes	Yes
SQLLite	Yes	No
SDTS	No	Yes
UK .NTF	No	Yes
U.S. Census TIGER/Line	No	Yes

2.1.1.2 Proj4

Proj4 is a coordinate re-projection library, capable of executing transformations between cartographic projection systems, and also between different spheroids and datums (where datum grid shifts are available).

The Proj4 library was originally written by Gerald Evenden as a utility library for the US Geological Survey (USGS). The current maintainer is Frank Warmerdam, who began maintaining Proj4 after Evenden ceased actively working on the project. Evenden remains active on the mailing list, and is currently providing new mathematical projections, though not providing code maintenance.

Maintainer: Frank Warmerdam (warmerdam@pobox.com)

Web Site: http://proj.maptools.org/

Implementation Language: C

Source License: MIT-style

Projections supported by the Proj4 library (projection code and common name):

aea : Albers Equal Area	mill : Miller Cylindrical
aeqd : Azimuthal Equidistant	mpoly : Modified Polyconic
airy : Airy	moll : Mollweide
aitoff : Aitoff	murd1 : Murdoch I
alsk : Mod. Stererographics of Alaska	murd2 : Murdoch II
apian : Apian Globular I	murd3 : Murdoch III
august : August Epicycloidal	nell : Nell
bacon : Bacon Globular	nell_h : Nell-Hammer
bipc : Bipolar conic of western hemisphere	nicol : Nicolosi Globular
boggs : Boggs Eumorphic	nsper : Near-sided perspective
bonne : Bonne (Werner lat_1=90)	nzmg : New Zealand Map Grid
cass : Cassini	ob_tran : General Oblique Transformation
cc : Central Cylindrical	ocea : Oblique Cylindrical Equal Area
cea : Equal Area Cylindrical	oea : Oblated Equal Area
chamb : Chamberlin Trimetric	omerc : Oblique Mercator
collg : Collignon	ortel : Ortelius Oval
crast : Craster Parabolic (Putnins P4)	ortho : Orthographic
denoy : Denoyer Semi-Elliptical	pconic : Perspective Conic
eck1 : Eckert I	poly : Polyconic (American)
eck2 : Eckert II	putp1 : Putnins P1
eck3 : Eckert III	putp2 : Putnins P2
eck4 : Eckert IV	putp3 : Putnins P3
eck5 : Eckert V	putp3p : Putnins P3'
eck6 : Eckert VI	putp4p : Putnins P4'
eqc : Equidistant Cylindrical (Plate Caree)	putp5 : Putnins P5
eqdc : Equidistant Conic	putp5p : Putnins P5'
euler : Euler	putp6 : Putnins P6
fahey : Fahey	putp6p : Putnins P6'
fouc : Foucaut	qua_aut : Quartic Authalic
fouc_s : Foucaut Sinusoidal	robin : Robinson
gall : Gall (Gall Stereographic)	rpoly : Rectangular Polyconic
gins8 : Ginsburg VIII (TsNIIGAiK)	sinu : Sinusoidal (Sanson-Flamsteed)
gn_sinu : General Sinusoidal Series	somerc : Swiss. Obl. Mercator
gnom : Gnomonic	stere : Stereographic
goode : Goode Homolosine	tcc : Transverse Central Cylindrical
gs48 : Mod. Stererographics of 48 U.S.	tcea : Transverse Cylindrical Equal Area
gs50 : Mod. Stererographics of 50 U.S.	tissot : Tissot
hammer : Hammer & Eckert-Greifendorff	tmerc : Transverse Mercator
hatano : Hatano Asymmetrical Equal Area	tpeqd : Two Point Equidistant
imw_p : International Map of the World Polyconic	tpers : Tilted perspective
kav5 : Kavraisky V	ups : Universal Polar Stereographic
kav7 : Kavraisky VII	urm5 : Urmaev V
labrd : Laborde	urmfps : Urmaev Flat-Polar Sinusoidal
laea : Lambert Azimuthal Equal Area	utm : Universal Transverse Mercator (UTM)
lagrng : Lagrange	vandg : van der Grinten (I)
larr : Larrivee	vandg2 : van der Grinten II
lask : Laskowski	vandg3 : van der Grinten III
latlong : Lat/long (Geodetic)	vandg4 : van der Grinten IV
longlat : Lat/long (Geodetic)	vitk1 : Vitkovsky I
lcc : Lambert Conformal Conic	wag1 : Wagner I (Kavraisky VI)
leac : Lambert Equal Area Conic	wag2 : Wagner II
lee_os : Lee Oblated Stereographic	wag3 : Wagner III
loxim : Loximuthal	wag4 : Wagner IV
lsat : Space oblique for LANDSAT	wag5 : Wagner V
mbt_s : McBryde-Thomas Flat-Polar Sine (No. 1)	wag6 : Wagner VI
mbtfpp : McBride-Thomas Flat-Polar Parabolic	wag7 : Wagner VII
mbtfpq : McBryde-Thomas Flat-Polar Quartic	weren : Werenskiold I
mbtfps : McBryde-Thomas Flat-Polar Sinusoidal	wink1 : Winkel I
merc : Mercator	wink2 : Winkel II
mil_os : Miller Oblated Stereographic	wintri : Winkel Tripel

2.1.1.3 GEOS

GEOS is the "Geometry Engine, Open Source", a C++ implementation of the JTS topology library. GEOS provides C++ implementations of all the simple features objects found in the OpenGIS "Simple Features for SQL" specification, and implementations of all the methods defined for those objects.

Topological calculations are easy to visualize, but hard to implement in generality. The GEOS/JTS algorithms are robust for all the spatial predicates (geometric comparisons which return true/false values). The GEOS/JTS algorithms are also strong in the spatial operators (geometric functions which produce geometric results).

Some Important GEOS Methods	
Predicates	**Operators**
Relate(Geom) Touches(Geom) Disjoint(Geom) Intersects(Geom) Contains(Geom) Crosses(Geom) Within(Geom) Overlaps(Geom) IsValid() IsSimply() IsRing()	Intersection(Geom) Union(Geom) Difference(Geom) Buffer(Tolerance) Distance(Geom) Length() Area() Perimeter() AsBinary() AsText()

Maintainer: Refractions Research (info@refractions.net)

Web Site: http://geos.refractions.net/

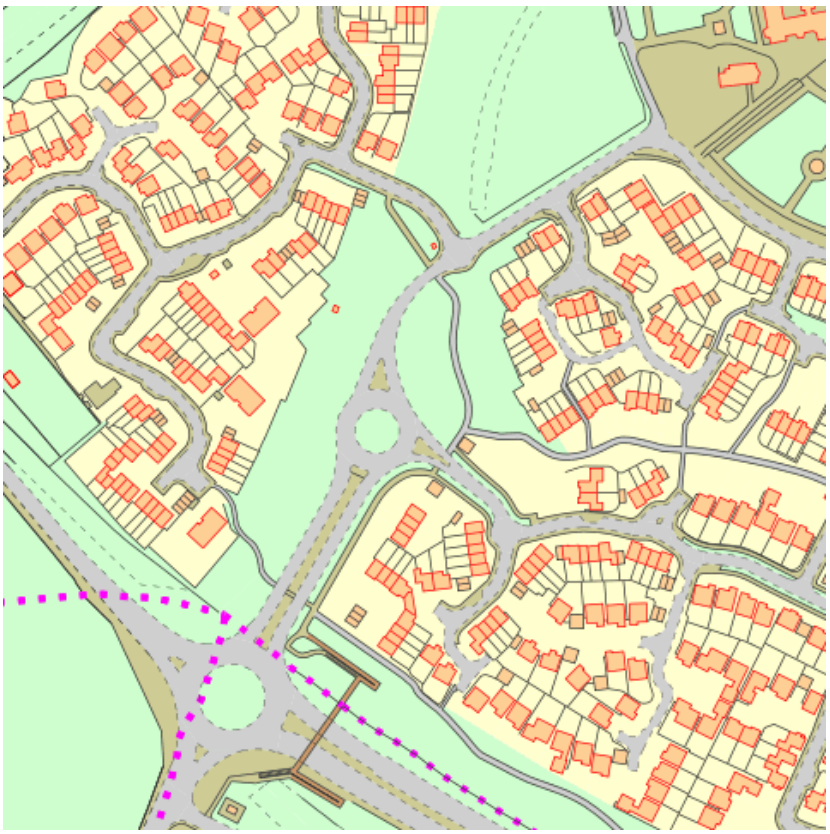
Implementation Language: C++

Source License: LGPL

2.1.1.4 Mapnik

Mapnik is a recent project, with a very small development team and user base at this point, but it holds some substantial promise. Mapnik appears to have a genesis in a developer reviewing the architecture of Mapserver and deciding to "do it right". The result is a C++ library built with a different set of dependencies (AGG for rendering instead of GD, C++ and associated Boost libraries instead of C) and an architecture thought out to be more extendible over the long term.

Thus far, Mapnik is still a work in progress, but already is producing some very fine cartographic output, and has some preliminary OpenGIS service implementations built on top of it.



Maintainer: Artem Pavlenko

Web Site: http://www.mapnik.org

Implementation Language: C++

Source License: LGPL

2.1.1.5 FDO

FDO stands for "Feature Data Objects", and is a data access library originally written by Autodesk for use in the Mapguide and Autodesk Map3D product lines. When MapGuide was open sourced in 2006, Autodesk also open sourced the supporting FDO library.

FDO is similar to OGR/GDAL, in providing a multi-format programming interface to different GIS vector and raster formats. However, FDO is more tuned to granular read/write operations than OGR, which makes it suitable for interactive applications like Map3D. FDO also includes concepts of locking, layers, and access control that do not appear in OGR.

FDO is generally a more complex library to use than OGR, because of the extra flexibility of the programming interface. It has fewer directly supported formats, but can also use OGR as a gateway to read all the OGR formats.

Format	Notes
SDF	File-based geodatabase format from Autodesk.
ArcSDE	Requires SDE client libraries.
MySQL	
ODBC	For non-spatial tabular data and x/y column data.
Oracle	Third-party Oracle native driver.
OGR	Read-only access to all OGR formats.
GDAL	
WMS	
WFS	
PostGIS	Third-party PostGIS native driver. Released 2007, still early release.

Maintainer: FDO Steering Committee

Web Site: http://fdo.osgeo.org/

Implementation Language: C++

Source License: LGPL

2.1.2 Applications

The C family of applications is a mixture of server-side applications and client-side applications, analytical tools and display tools. Most GIS workloads are covered in the application family, with the notable exception of paper map-making, the most common GIS workload.

Note: The saturated commercial market for cartography tools, the high level of effort to achieve usable tools, and the appeal of other cutting edge projects have combined to deter any active development on user-friendly paper map production tools. As with the OpenOffice experience in Linux, it would probably require a dedicated multi-year funded project to produce a core product with sufficient technical mass that an open source community could reasonably continue with enhancements and support.

2.1.2.1 MapGuide Open Source

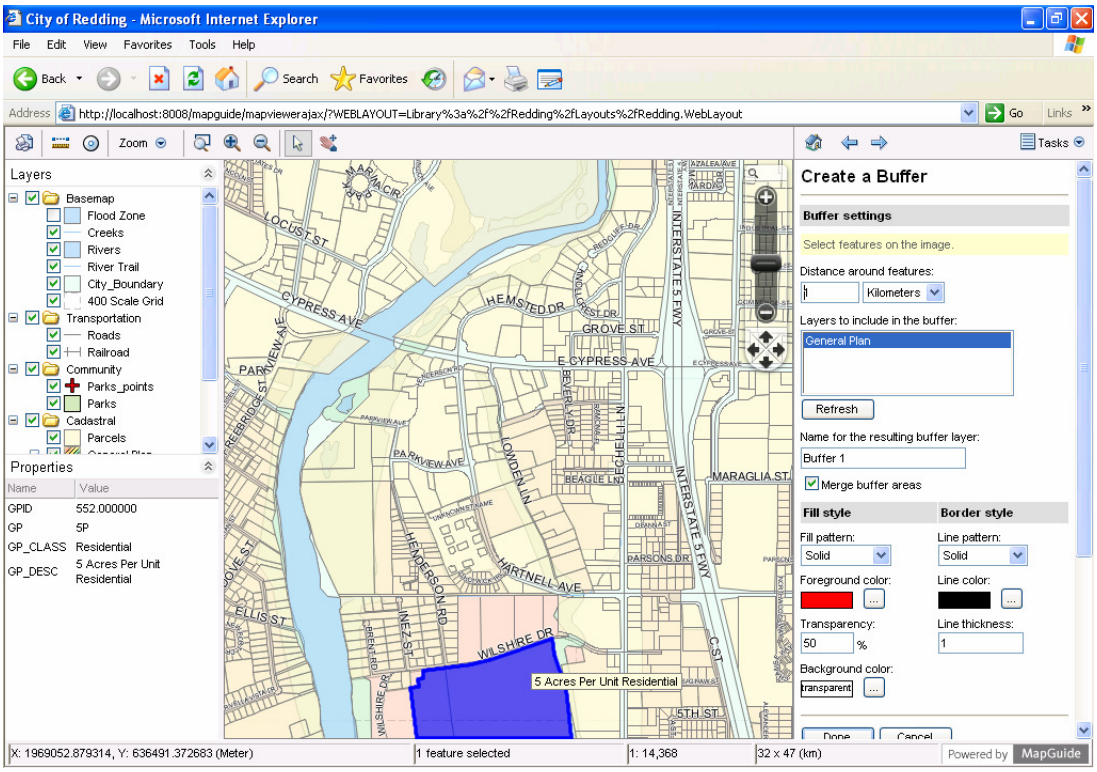
Despite sharing a name with the previous closed source MapGuide product from Autodesk, MapGuide Open Source (OS) is in fact a completely new product, with a new code base and a new licensing philosophy. Autodesk will sell the new MapGuide as commercial product, with some bonus features (extra format support, formal product support, better backward compatibility) but the main development of the MapGuide OS product is now done as open source. (The situation is similar to the arrangement Netscape/AOL had with the Mozilla web browser some years ago – bundling a separate closed source version based on an open source core.)

From their web site:

> *MapGuide Open Source is a web-based platform that enables users to quickly develop and deploy web mapping applications and geospatial web services. MapGuide features an interactive viewer that includes support for feature selection, property inspection, map tips, and operations such as buffer, select within, and measure. MapGuide includes an XML database for managing content, and supports most popular geospatial file formats, databases, and standards. MapGuide can be deployed on Linux or Windows, supports Apache and IIS web servers, and offers extensive PHP, .NET, Java, and JavaScript APIs for application development. MapGuide Open Source is licensed under the LGPL.*

As a more recent project, MapGuide has a more modern architecture than the original MapServer. It also includes some default web interface components as well, so it is possible to create an out-of-the-box web mapping site with MapGuide more easily than with Mapserver. Mapserver has its own advantages, in terms of simplicity and number of supported formats, so examining both carefully before making a decision is a good idea.

Because the originating organization is Autodesk, some users might be concerned that MapGuide OS is not "real" open source. However, it certainly is "real", judging from a number of facts. First, the license used is not some customized corporate license, but the familiar LGPL, used by many other open source projects. Second, like other open source projects, the new MapGuide OS code base includes dependencies on other open source library projects, such as Proj4 and GEOS – enlightened re-use is a sign of a good open source methodology. Finally, Autodesk has opened up the development process, using a public source code repository for active development, having a public mailing list for users and developers to directly interact, and transferring all intellectual property rights for the code to a neutral organization (the Open Source Geospatial Foundation).



Maintainer: Autodesk

Web Site: http://mapguide.osgeo.org/

Implementation Language: C++

Source License: LGPL

2.1.2.2 UMN Mapserver

The University of Minnesota MapServer (commonly called just "MapServer") is an internet map server, a server-side piece of software which renders GIS data sources into cartographic map products on-the-fly.

On OSS evaluation merits, MapServer is easily the most successful open source GIS project to date.

MapServer has a multi-disciplinary community, has core team members with 100% of their time devoted to product maintenance and enhancement, has an open core team, substantial documentation, and a transparent release process. The modularity of the project has been improved with each release, and now supports both multiple input format types and multiple output render types.

On technical merits, MapServer is also extremely successful. It supports more input data sources than most proprietary products, has higher performance, and (in the precompiled versions) is simpler to install and set up.

Input Formats	**Output Formats**	**API Access**
Shape	GIF	MapServer CGI
PostgreSQL	JPEG	MapScript Python
OracleSpatial	PNG	MapScript Perl
ArcSDE	All GDAL Formats	MapScript PHP
Remote WMS Layers	GML	MapScript Java
JPG/WRL	Flash	MapScript .Net
GIF/WRL	PDF	C API
PNG/WRL		OpenGIS WMS
All GDAL Formats		OpenGIS WFS
All OGR Formats		OpenGIS WCS
		OpenGIS SOS

Maintainer: MapServer Core Team (mapserver-dev@lists.gis.umn.edu)

Web Site: http://mapserver.gis.umn.edu

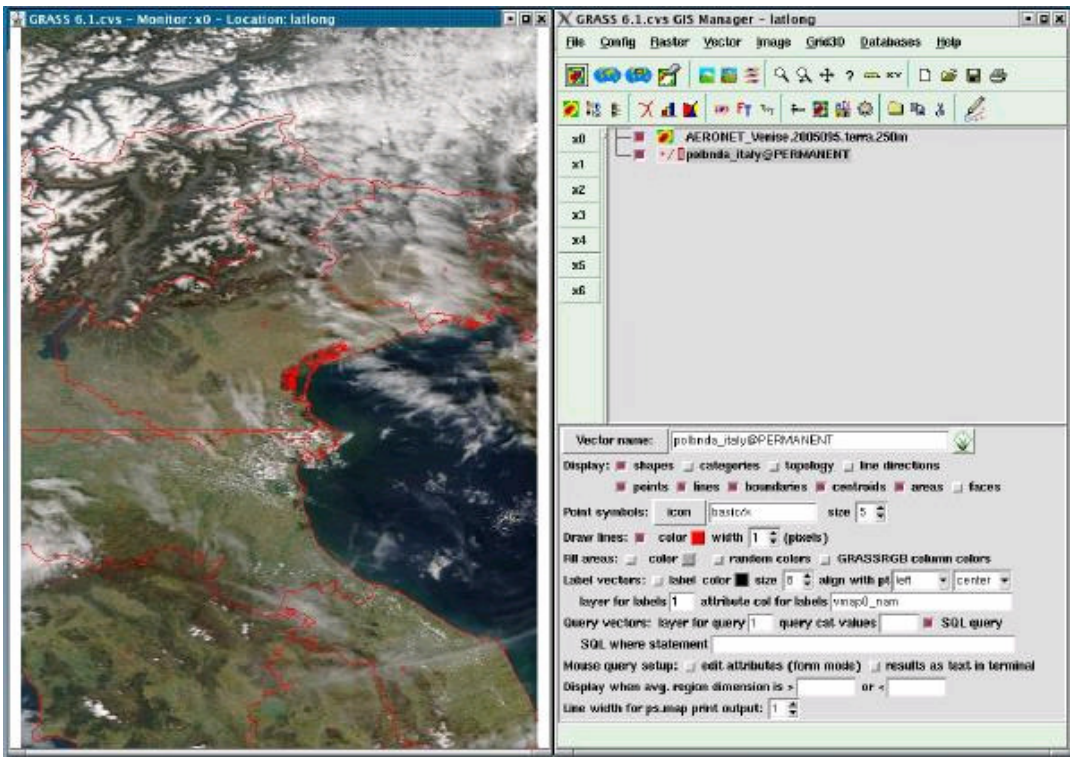
Implementation Language: C

Source License: MIT-style

2.1.2.3 GRASS

GRASS is easily the oldest of the open source GIS software products. It was originally a closed project of the US Army, started in 1982 to provide capabilities that did not exist in the commercial GIS sector. The Army maintained GRASS under active development until 1992, and continued with fixes and patches through 1995. GRASS was picked up by the academic community in 1997, when Baylor University began coordinating development, and was officially "open sourced" in 1999 under the GPL. Since 2001, the GRASS project has been headquartered at ITC, in Trento, Italy.

Originally written as a raster analysis system, GRASS has had vector analysis capabilities added to it as well. GRASS can import a wide range of formats, using both the GDAL and OGR libraries for data import. GRASS also has the ability to directly read attribute and spatial data from PostGIS/PostgreSQL.



GRASS has been most historically effective as a modeling tool, carrying out complex data analysis tasks. The list of applications at the GRASS home page (http://grass.itc.it/applications/index.php) gives a flavor of the kinds of problems GRASS is being used to solve.

Maintainer: GRASS Development Team

Web Site: http://grass.itc.it/

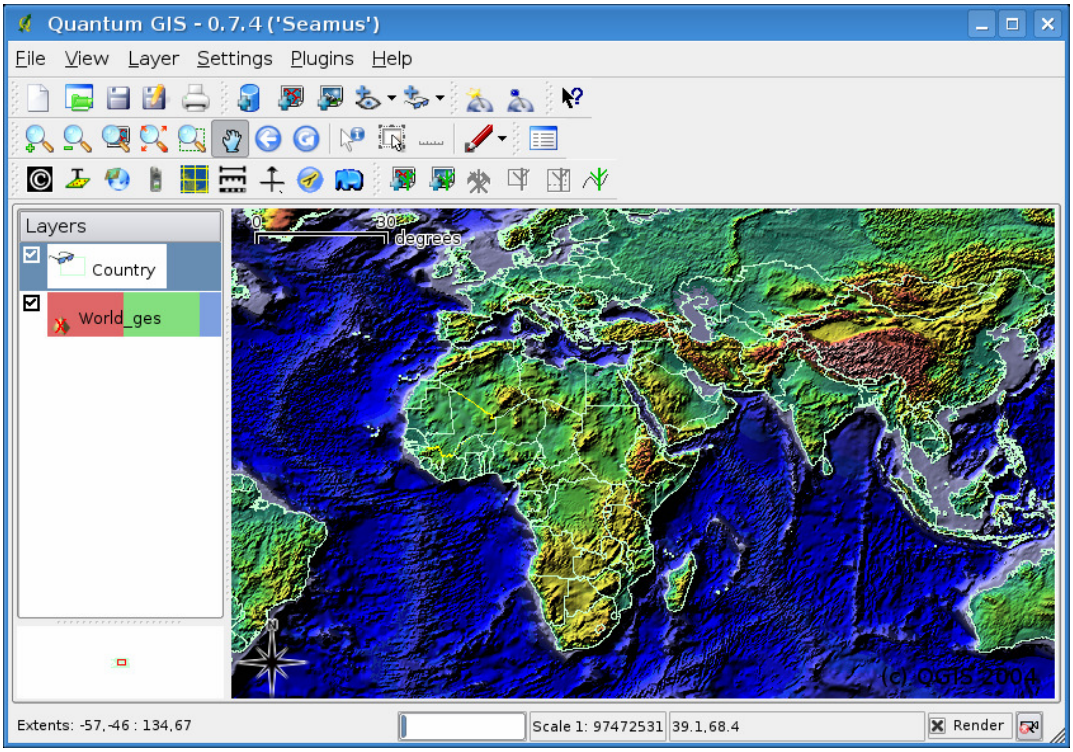
Implementation Language: C

Source License: GPL

2.1.2.4 QGIS

QGIS is a GIS viewing environment built primarily for the Linux desktop. QGIS depends on the QT widget set, which is a same widget set used by the popular KDE desktop environment. However, QT is available for other platforms (Win32, OS/X, Solaris) so a QGIS desktop can be built for use in a multi-platform environment.

QGIS supports PostGIS and Shapefiles as vector data sources. QGIS uses OGR as a data import bridge, so support of all OGR formats is also available. QGIS supports DEM, ArcGrid, ERDAS, SDTS, and GeoTIFF raster formats.



QGIS has increased in development tempo in 2004, completing several minor releases and adding important new features with each release, including a recent tie in to GRASS analysis functionality. The developer community is now rapidly increasing beyond the original founder.

Maintainer: Gary Sherman (gsherman@sourceforge.net)

Web Site: http://www.qgis.org/

Implementation Language: C++

Source License: GPL

2.1.2.5 OSSIM

OSSIM (Open Source Software Image Map) is a raster manipulation tool chain. OSSIM is primarily developed by Intelligence Data Systems and used by that company for many cutting-edge image processing projects in the US government. Sanz also uses OSSIM in their EarthWhere product line of high end raster storage and manipulation appliances.

OSSIM is a C++ library, with a number of applications built on top. The primary technical benefit of OSSIM is that it is architected to cut image processing tasks into independent and parallelizable components. As a result, OSSIM-based processing tasks can be run on high performance computing arrays, such as Beowulf clusters, for massive performance increases.

OSSIM processing streams are built up as "task chains", tying together different processing modules to turn raw imagery into completed product.

Maintainer: Radiant Blue Technologies

Web Site: http://www.ossim.org

Implementation Language: C++

Source License: GPL

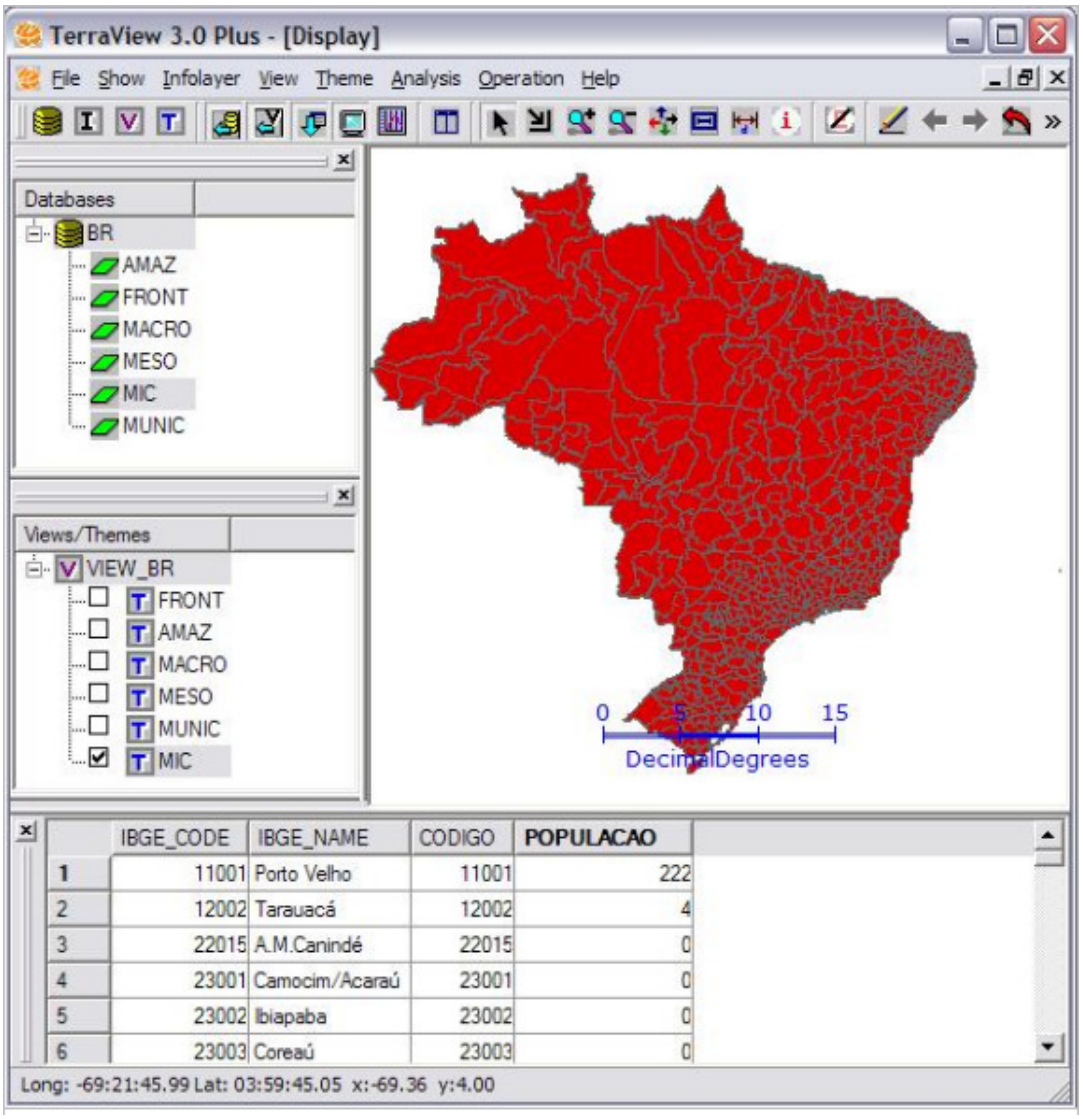
2.1.2.6 TerraLib

TerraLib is a GIS classes and functions library written in C++, developed by a branch of the Brazilian space agency (INPE) for use throughout the Brazilian government. TerraLib is the second generation of development, based on the original "Spring GIS" project. Its main aim is to enable the development of a new generation of GIS applications, based on the technological advances on spatial databases.

TerraLib implements the following data and service modules:

- Metadata Model
- Spatial Model
- Spatial Model to Oracle Spatial
- Spatial Model to PostgresSQL
- Spatial Model to PostGIS
- Application Model
- GeoCoding Model
- Spatial Statistics Model

The documentation and other information about TerraLib is now generally available in both Portuguese and English, and the developers answer questions on the web forums in English. TerraLib includes a viewer/editor package, TerraView, that uses the library as a data access/manipulation layer.



Maintainer: INPE (Brazil)

Web Site: http://www.terralib.org/

Implementation Language: C++

Source License: LGPL

2.1.2.7 GMT

The "Generic Mapping Tools" (GMT) is a project with a very long history. Developed in an academic environment in the University of Hawaii since 1988, GMT is designed as a suite of small data manipulation and graphic generation programs, that can be sequenced and scripted together to create complex data processing chains. For example, GMT applications can take raw data in from sensors, create an interpolated grid, contour the grid, and create plotter-ready files for printing in automated batch streams.

FILTERING OF 1-D AND 2-D DATA:

blockmean L2 (x,y,z) data filter/decimator
blockmedian L1 (x,y,z) data filter/decimator
blockmode Mode-estimating (x,y,z) data filter/decimator
filter1d Filter 1-D data (time series)
grdfilter Filter 2-D data in space domain

PLOTTING OF 1-D and 2-D DATA:

grdcontour Contouring of 2-D gridded data
grdimage Produce images from 2-D gridded datar
grdvector Plot vector fields from 2-D gridded data
grdview 3-D perspective imaging of 2-D gridded data
psbasemap Create a basemap frame
psclip Use polygon files as clipping paths
pscoast Plot coastlines, filled continents, rivers, and political borders
pscontour Direct contouring or imaging of xyz-data by triangulation
pshistogram Plot a histogram
psimage Plot Sun rasterfiles on a map
psmask Create overlay to mask specified regions of a map
psrose Plot sector or rose diagrams
psscale Plot grayscale or colorscale
pstext Plot textstrings
pswiggle Draw anomalies along track
psxy Plot symbols, polygons, and lines in 2-D
psxyz Plot symbols, polygons, and lines in 3-D

GRIDDING OF (X,Y,Z) DATA:

nearneighbor Nearest-neighbor gridding scheme
surface Continuous curvature gridding algorithm
triangulate Perform optimal Delauney triangulation on xyz data

SAMPLING OF 1-D AND 2-D DATA:

grdsample Resample a 2-D gridded data onto new grid
grdtrack Sampling of 2-D data along 1-D track
sample1d Resampling of 1-D data

PROJECTION AND MAP-TRANSFORMATION:

grdproject Project gridded data onto new coordinate system
mapproject Transformation of coordinate systems
project Project data onto lines/great circles

INFORMATION:

gmtdefaults List the current default settings
gmtset Edit parameters in the .gmtdefaults file
grdinfo Get information about grd files
minmax Report extreme values in table datafiles

CONVERT OR EXTRACT SUBSETS OF DATA:

gmtconvert Convert table data from one format to another
gmtmath Reverse Polish calculator for table data
gmtselect Select table subsets based on multiple spatial criteria
grd2xyz Convert 2-D gridded data to table
grdcut Cut a sub-region from a grd file
grdpaste Paste together grdfiles along common edge
grdreformat Convert from one grdformat to another
splitxyz Split xyz files into several segments
xyz2grd Convert table to 2-D grd file

MISCELLANEOUS:

makecpt Create GMT color palette tables
spectrum1d Compute spectral estimates from time-series
triangulate Perform optimal Delauney triangulation on xyz data

DETERMINE TRENDS IN 1-D AND 2-D DATA:

fitcircle Finds best-fitting great or small circles
grdtrend Fits polynomial trends to grdfiles (z = f(x,y))
trend1d Fits polynomial or Fourier trends to y = f(x) series
trend2d Fits polynomial trends to z = f(x,y) series

OTHER OPERATIONS ON 2-D GRIDS:

grd2cpt Make color palette table from grdfile
grdclip Limit the z-range in gridded data sets
grdedit Modify grd header information
grdfft Operate on grdfiles in frequency domain
grdgradient Compute directional gradient from grdfiles
grdhisteq Histogram equalization for grdfiles
grdlandmask Creates mask grdfile from coastline database
grdmask Set nodes outside a clip path to a constant
grdmath Reverse Polish calculator for grdfiles
grdvolume Calculating volume under a surface within a contour

Maintainer: Paul Wessel & Walter Smith

Web Site: http://gmt.soest.hawaii.edu/

Implementation Language: C

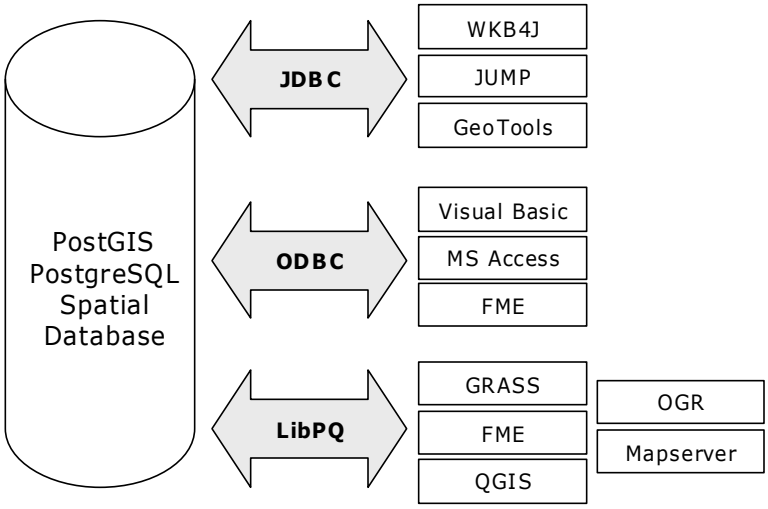
Source License: GPL

2.1.2.8 PostGIS

PostGIS adds spatial database capabilities to the PostgreSQL (www.postgresql.org) object-relational database. The PostGIS extension adds:

- Proper spatial objects (point, line, polygon, multipoint, multiline, multipolygon, geometrycollection)
- Spatial indexing (r-tree)
- Simple analytical functions (area, length, distance)
- Predicates (via GEOS)
- Operators (via GEOS)
- Coordinate system metadata
- Coordinate reprojection support (via Proj4)
- Data import and export tools

The strength of PostGIS is that it has become the standard spatial database backend for all the other open source GIS tools. As a result, a layer in PostGIS can be analyzed with GRASS, published over the web with Mapserver and MapGuide, visualized on the desktop with uDig, gvSIG or QGIS, and exported to proprietary formats with OGR and FDO.



PostGIS is also used heavily by applications and libraries in the Java development language, via the standard JDBC (Java Database Connectivity) libraries.

Maintainer: Refractions Research Inc

Web Site: http://postgis.refractions.net

Implementation Language: C

Source License: GPL

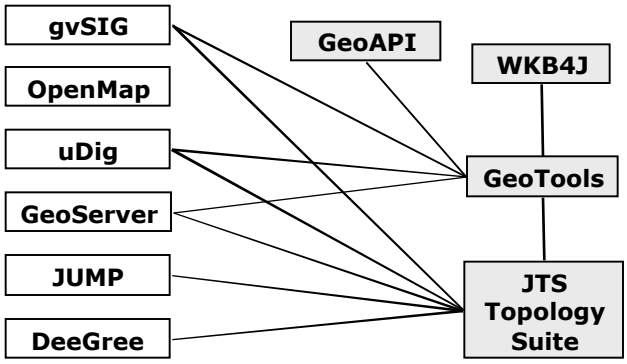
2.2 Survey of 'Java' Projects

The "Java" world includes several independent attempts at "complete unified toolkits" – OpenMap, GeoTools, and deegree. In addition, two of the desktop applications, JUMP and gvSIG, have considerable independent implementations of core features like data access and rendering, though they do use GeoTools library code for some functionality (coordinate re-projection primarily).

The GeoTools library has been used as a building block for two applications, the uDig desktop application, and the GeoServer web services application. In fact, the two applications share significant functionality via GeoTools: re-projection, rendering, data access, and styling to name a few.

Almost all the Java projects (OpenMap excepted) use the JTS library as either their core geometry representation, or as a utility class for geometric operations.

As a result, development in the Java world is currently concentrated around projects that use the JTS Topology Suite as the basis for geometry representation, with a secondary node growing around the GeoTools library.



Side projects, such as GeoAPI (interface standards) are also used either directly by the projects or by applications that use the toolkit chain.

2.2.1 Shared Libraries

2.2.1.1 JTS Topology Suite

JTS is the central geometry library for much of the ongoing Java GIS development. JTS provides a Java implementation of the OpenGIS "Simple Features Specification", in particular the functions described in the "Simple Features for SQL Specification".

The element that makes JTS special is the implementation of the "spatial predicates". Spatial predicates are functions which compare two spatial objects and return a boolean true/false result indicating the existence (or absence) of a particular spatial relationship. Some examples of spatial predicates are Contains(), Intersects(), Touches(), and Crosses(). The JTS implementation of the predicates is special in that the functions are all "robust" – that is, there is no special case of strange geometries or odd coordinates which is capable of producing a failure or incorrect result. This is a unique property – most proprietary products do not include robust spatial predicates.

JTS also includes implementations of the spatial "operators" which take two geometries and return a new derived geometric result. Examples of the operators include Difference(), Union(), and Buffer(). The JTS operator implementations have been widely tested, but do not have robustness guarantees like the predicates. Each successive release of JTS improves operator robustness and removes more failure cases.

Spatial predicate and operator implementations are valuable because they are extremely difficult to code. For this reason, the JTS library is widely reused by other OSS projects. By using JTS, they get a standard set of geometries, with the most difficult spatial methods already implemented. GEOS, a port of JTS to C++, serves this same role for the C family of applications.

Maintainer: Martin Davis (mbdavis@refractions.net)

Web Site: http://www.jump-project.org/

Implementation Language: Java

Source License: LGPL

JTS development was originally funded by GeoConnections (www.geoconnections.org).

2.2.1.2 GeoTools

GeoTools is an open source, Java GIS toolkit for developing OpenGIS- and ISO-compliant solutions. It has a modular architecture that allows extra functionality to be added or removed easily. GeoTools aims to support OpenGIS and other relevant standards as they are developed.

The aim of the project is to develop a core set of Java objects in a framework which makes it easy for others to implement OGC compliant server-side services or provide OGC compatibility in standalone applications or applets. The GeoTools project comprises a core API of interfaces and default implementations of those interfaces.

It is not the intention of the GeoTools project to develop finished products or applications, but it is the intention to interact and fully support other initiatives and projects which would like to use the GeoTools 2 toolkit to create such resources.

GeoTools features and goals:

GeoTools code is built using the latest Java tools and environments (Java 1.5 at time of writing) and will continue to leverage the capabilities of future Java environments and official extensions as and when the technologies are released and have been through the first maintenance cycle (i.e. version 1.x.1)

GeoTools is being built as modularly as possible in a way that allows interested parties to use the functionality that they are interested in without needing to know about or include the functionality that they are not interested in.

Modules are built which support individual OGC specifications (e.g. Filter, SLD, GML2) and which also support interaction with a wide range of data-sources (e.g Shapefile, MIF/MID, ArcSDE, Oracle, PostGIS and MySQL). Modules each have their own maintainers who control the content and direction of that module. The GeoTools project actively encourages suggestions for new modules and invites interested developers to start new modules for new functionality or to help drive and develop existing modules.

The overall maintenance and future direction of GeoTools are managed by the GeoTools Project Management Committee. Currently this comprises 7 active developers who take joint responsibility for design and implementation decisions. The team welcomes and encourages others to become contributors and ultimately become part of the GeoTools development team.

It is a long-term goal of the GeoTools project to refine its core API and promote its use so that it can become a recognized and standard API for geospatial development.

Maintainer: GeoTools Project Management Committee

Web Site: http://www.geotools.org

Implementation Language: Java

Source License: LGPL

2.2.2 Applications

2.2.2.1 GeoServer

The GeoServer project is a Java (J2EE) implementation of the OpenGIS Consortium's web services specification – Web Map Server (WMS) and Web Feature Server (WFS). It is free software, available under the GPL 2.0 license.

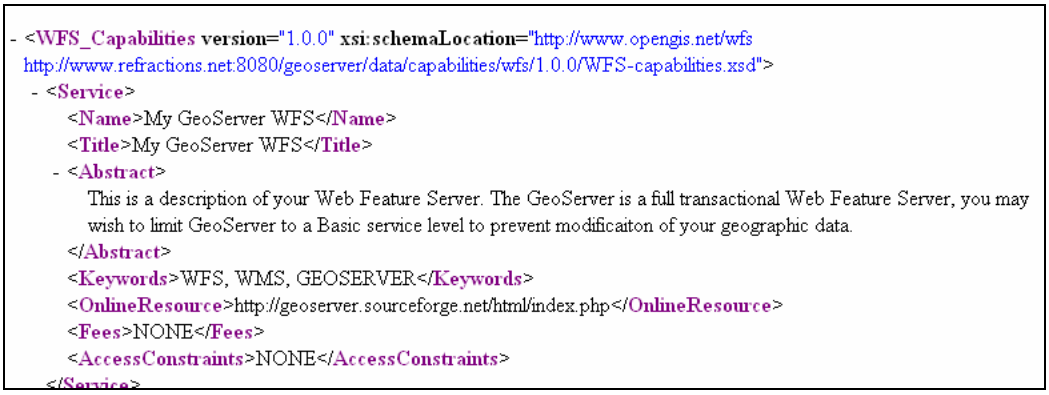
GeoServer is built on top of the GeoTools library, and as a result, much of the internal logic of the server (data sources, GML parsing, XML Filter support, etc) actually resides and is maintained at the GeoTools library level. In this respect, it is best to consider the two projects as conjoined entities – GeoServer/GeoTools. (The uDig desktop project has a similarly close relationship to GeoTools.)

The GeoServer WFS has been chosen by OpenGIS as a reference implementation for use in the OpenGIS "CITE" interoperability portal. As a reference implementation, GeoServer is required to support all aspects of the current and evolving specification.

GeoServer can currently serve WFS on top of:

- Oracle Spatial
- ArcSDE
- PostGIS
- ESRI Shape Files

In addition to WFS and WMS support, GeoServer includes support for KML output, tiled map output, and web-based configuration and management.



GeoServer passes all OpenGIS conformance tests and is fully compliant with the Web Feature Server 1.0 and 1.1 specifications.

Maintainer: The Open Planning Project (http://www.openplans.org)

Web Site: http://geoserver.org/

Implementation Language: Java

Source License: GPL

2.2.2.2 deegree

deegree (formerly known as "JaGo") was developed initially in an academic environment at the University of Bonn in Germany. The architecture is a message passing system, designed to be both extremely modular and highly de-coupled. The deegree architecture allows various components of the system to run on different machines while still presenting a unified system to the outside world.

Before leaving the academic world, deegree completed considerable OpenGIS feature support, including both WMS and WFS server implementations. Supported data sources include shape file, RDBMS and OpenGIS data formats (WKB and WKT). Catalog server support, grid coverage server support and others are either fully are partially complete.

The architecture that makes deegree unique also makes understanding the code hard for the neophyte – learning curves can be steep.

As part of the CITE project, the GeoTools and deegree teams are working to harmonize underlying data models (feature and geometry models) and to bring some of the deegree capabilities (such as WMS) into the GeoTools/GeoServer projects for use in CITE.

Maintainer: DeeGree Team (info@lat-lon.de)

Web Site: http://deegree.sourceforge.net/

Implementation Language: Java

Source License: LGPL

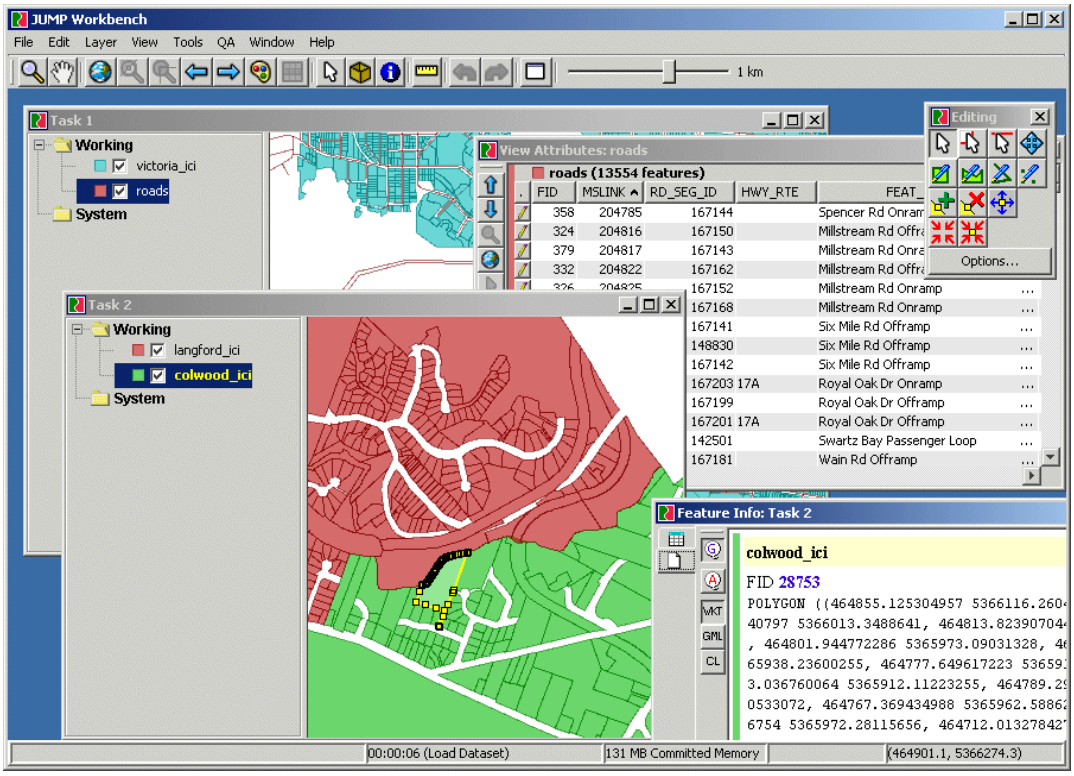
2.2.2.3 JUMP / OpenJUMP / Kosmo

JUMP is the "JUMP Unified Mapping Platform", a visualization and user interface toolkit originally written as a user interface wrapper for data integration algorithms.

JUMP was designed to be a generic and pluggable environment into which the complex algorithms required for spatial data conflation could be embedded. Spatial data conflation usually requires a human input element, and as a result JUMP was built with a number of generic user interface and GIS viewer features.

- JUMP provides an interactive Workbench for viewing, editing, and processing spatial datasets
- JUMP provides an API giving full programmatic access to all functions, including I/O, feature-based datasets, visualization, and all spatial operations
- JUMP is highly modular and extensible
- JUMP supports important industry standards such as GML and the OpenGIS Consortium spatial object model
- JUMP is written in 100% pure Java™.

JUMP supports GML, Shape, and RDBMS data sources.



JUMP has had an uneven development history, and has spawned a number of variant projects as a result. JUMP continues to be maintained in a semi-closed and slow moving process by the original development team – a new release is in the offing for 2006. The opacity of the core development process led to a parallel process, dubbed "OpenJUMP" (www.openjump.org) that quickly added multi-lingual support and numerous small interface improvements as well as some analytical plug-ins. OpenJUMP itself recently spawned a commercially developed and supported variant, Kosmo (http://www.saig.es/en/), in Spain.

Maintainer: JUMP - Vivid Solutions, http://www.vividsolutions.com/
Kosmo – SAIG, http://www.saig.es/
OpenJUMP – http://www.openjump.org/

Web Site: http://www.jump-project.org/

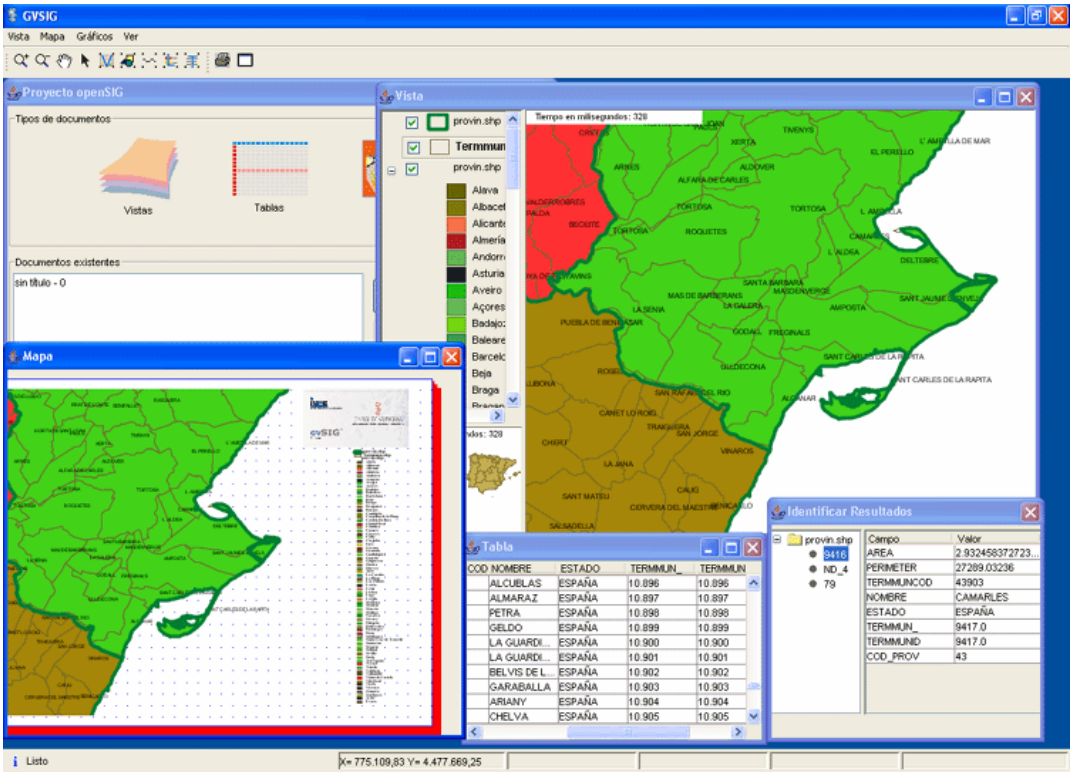
Implementation Language: Java

Source License: GPL

2.2.2.4 gvSIG

gvSIG is a project of the Spanish province of Valencia. The goals of the project are to provide an open source tool that utilizes open standards and is platform independent. gvSIG wraps a number of the Java libraries, including GeoTools and JTS.

The design goals of gvSIG are: modularity, interoperability, open source, standards based, low cost of deployment, and portability to multiple platforms.



As a desktop environment, gvSIG now includes all the core functionality you would expect from a viewer/editor: styling, selections, printing with layouts, data editing, raster and vector data support.

Maintainer: Valencia, Spain

Web Site: http://www.gvsig.gva.es/

Implementation Language: Java

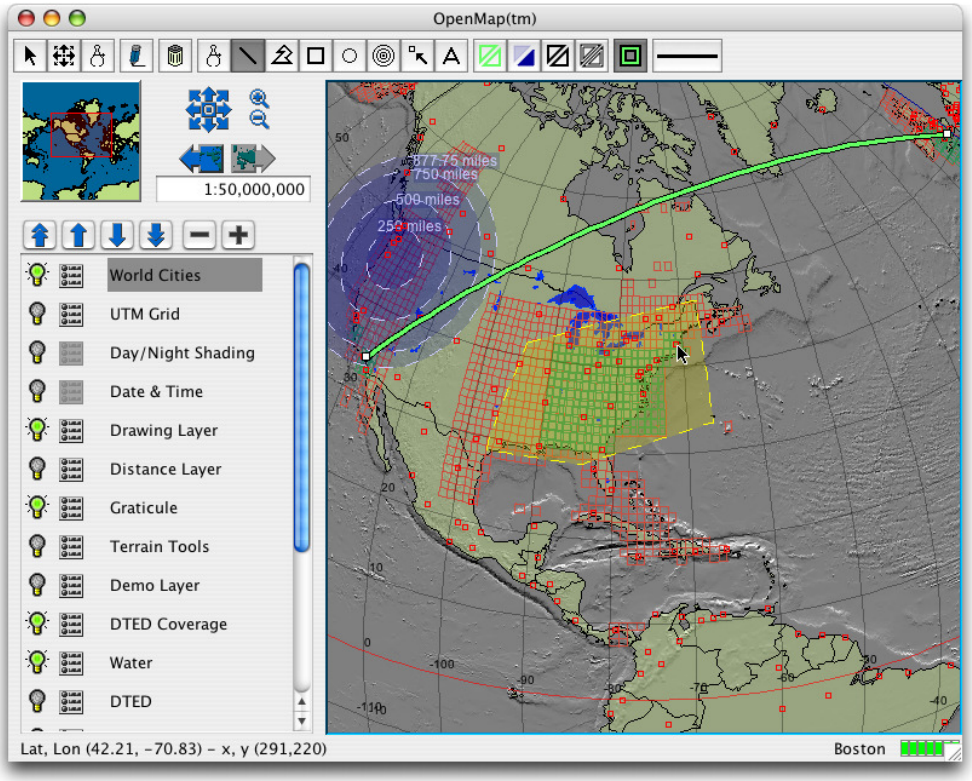
Source License: GPL

2.2.2.5 OpenMap

OpenMap is a component library for building spatial applications in Java. It was originally developed by BBN technologies for consulting projects with utility and telephony companies. It was the earliest open source Java spatial toolkit, and the code base is a little crusty at this point. The old architecture largely remains, but several new concepts and ways of accessing data have been overlain on top of it.

OpenMap is still being actively developed by BBN, who provides support contracts for companies that want to use OpenMap as part of a product or other deployment.

OpenMap supports Shapefiles as an input data source, but other data sources are largely coded from scratch. The "Layer" concept in OpenMap is sufficiently general that almost any data source can be slaved into an OpenMap application – for example, OpenMap ships with an example "EarthQuakes" layer which continuously updates against a public earthquake information HTML page to provide an always-current map of recent earthquakes.



Maintainer: BBN Technologies (openmap@bbn.com)

Web Site: http://openmap.bbn.com/

Implementation Language: Java

Source License: Mozilla-style

2.2.2.6 uDig

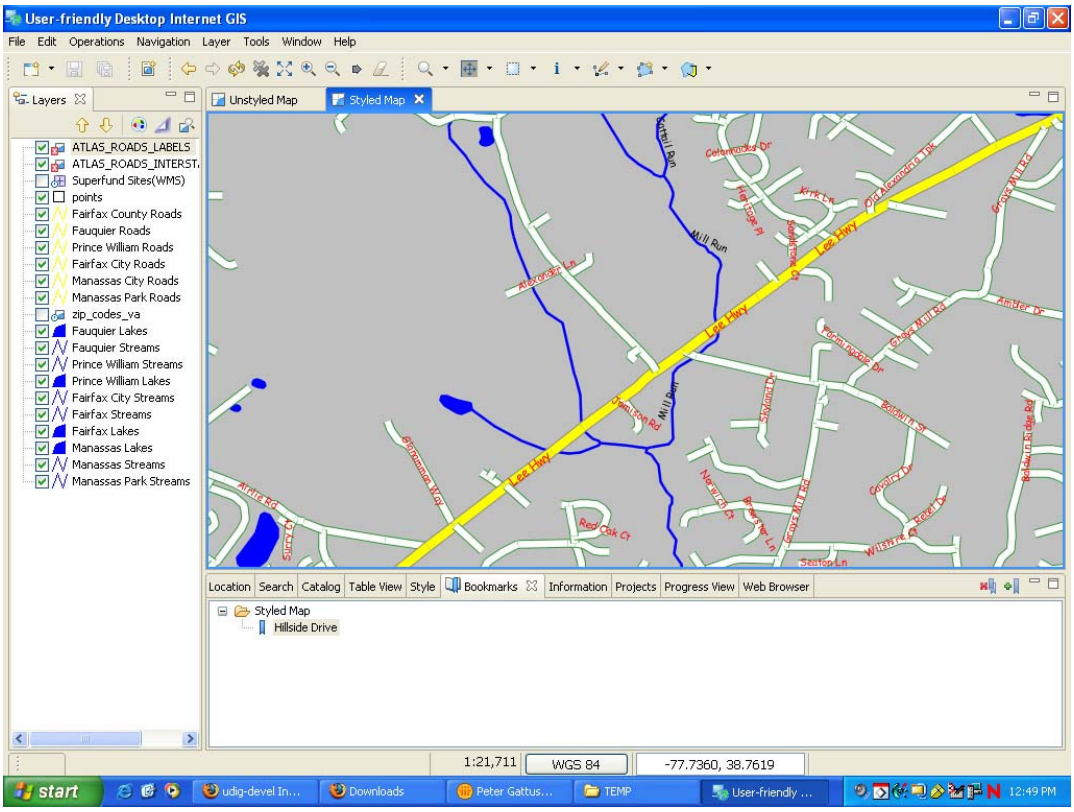
uDig is a project to join the strengths of the GeoTools project (design, data structures, standards) with the strengths of the JUMP project (UI, renderer, interactivity), and the strengths of the Eclipse Rich Client Platform (extensibility, industry development standards) into a new desktop editor capable of interacting with a range of local, network, and internet data sources.

uDig stands for "**U**ser-friendly **D**esktop **I**nternet **GIS**", and the goal is to bring internet mapping technologies such as WMS and WFS transparently to ordinary GIS users desktops.

uDig has the following capabilities:

- **WFS client read/write support**, to allow direct editing of data exposed via transactional Web Feature Servers (WFS-T).
- **WMS support**, to allow viewing of background data published via WMS.
- **Styled Layer Descriptor (SLD) support**, to allow the client-directed dynamic re-styling of WMS layers.
- **Web Catalog Server framework**, for quick search and use of online mapping services.
- **Printing support**, to allow users to create standard and large format cartography from their desktops.
- **Standard GIS file format support**, to allow users to directly open, overlay, and edit local Shape and GeoTIFF files with online data.
- **Coordinate projection support**, to transparently integrate remote layers in the client application where necessary.
- **Database access support**, to allow users to directly open, overlay and edit data stored in PostGIS, OracleSpatial, ArcSDE, and MySQL.
- **Cross-platform support**, using Java as an implementation language, and providing one-click setup files for Windows, OS/X, and Linux.
- **Multi-lingual design**, allowing easy internationalization of the interface, with French and English translations of the interface completed initially.
- **Customizability and modularity**, to allow third party developers to add new capabilities, or strip out existing capabilities as necessary when integrating the application with existing enterprise infrastructures.

The screen snap below shows uDig viewing WMS imagery, WFS data, and searching a remote catalogue server for information.



Maintainer: Refractions Research (info@refractions.net)

Web Site: http://udig.refractions.net/

Implementation Language: Java

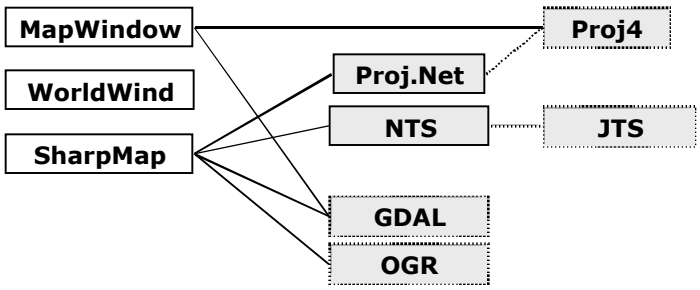
Source License: LGPL

2.3 Survey of .Net Projects

The family of .Net projects is still relatively small, and the inter-linkages are only just being made. However, by building on the library basis of the Java and C communities, the .Net community has laid a solid base to build upon.

The NTS and Proj.Net libraries are ports of JTS and Proj4, respectively, providing topological operations and coordinate transformation – the core of any GIS system.

On top of those libraries, the SharpMap project is building a rendering, styling and analytical library, suitable for building into yet larger and more complex applications. SharpMap is already being used in web services, and a desktop version cannot be far behind.



An interesting quirk of the .Net tribe is that it can use C++ library code directly, without porting to C#, by wrapping it into a .Net assembly. As a result, both SharpMap and MapWindow are able to make use of the extensive format support in the GDAL/OGR libraries from the C tribe.

2.3.1 Libraries

2.3.1.1 NTS

The Net Topology Suite (NTS) is a C# port of the JTS Topology Suite from Java. Because the languages are quite similar, the NTS is kept well synchronized to changes in JTS. NTS provides all the features that JTS does, but in a form suitable for direct use in .Net.

Maintainer: Diego Guidi

Web Site: http://code.google.com/p/nettopologysuite/

Implementation Language: C#

Source License: LGPL

2.3.1.2 Proj.Net

Proj.Net is a C# coordinate projection library. Calling it a "port" of Proj4 is not strictly accurate, since the Proj.Net library is built using more object-oriented structures, and is not a simple 1:1 mapping of the Proj4 library. However, Proj.Net does use the algorithms from Proj4 for the actual mathematics, so the effort of development Proj.Net is reduced to writing the language implementation, not figuring out the tricky mathematically algorithms.

Maintainer: Deigo Guidi

Web Site: http://www.codeplex.com/ProjNET

Implementation Language: C#

Source License: LGPL

2.3.1.3 SharpMap

A .Net library for mapping applications, both internet and desktop, SharpMap builds on top of the NTS and Proj.Net libraries to create a full feature model, data access model, and rendering model.

SharpMap also makes use of OGR/GDAL to access data in the multiple formats supported by those libraries. SharpMap is receiving lots of attention as a quick and easy replacement for the ESRI MapObjects library. Like MO, it is a convenient toolkit for Windows-comfortable developers, and provides simple access to GIS primitives like feature access and map generation.

Street map rendered by SharpMap

SharpMap has a similar feature coverage to GeoTools, but is built with a simpler, and probably easier-to-grasp architecture model. In addition to the core infrastructure, the team is working on AJAX bindings, and desktop bindings for a more complete library that goes from data all the way to UI.

Maintainer: Rory Plaire

Web Site: http://www.codeplex.com/SharpMap

Implementation Language: C#

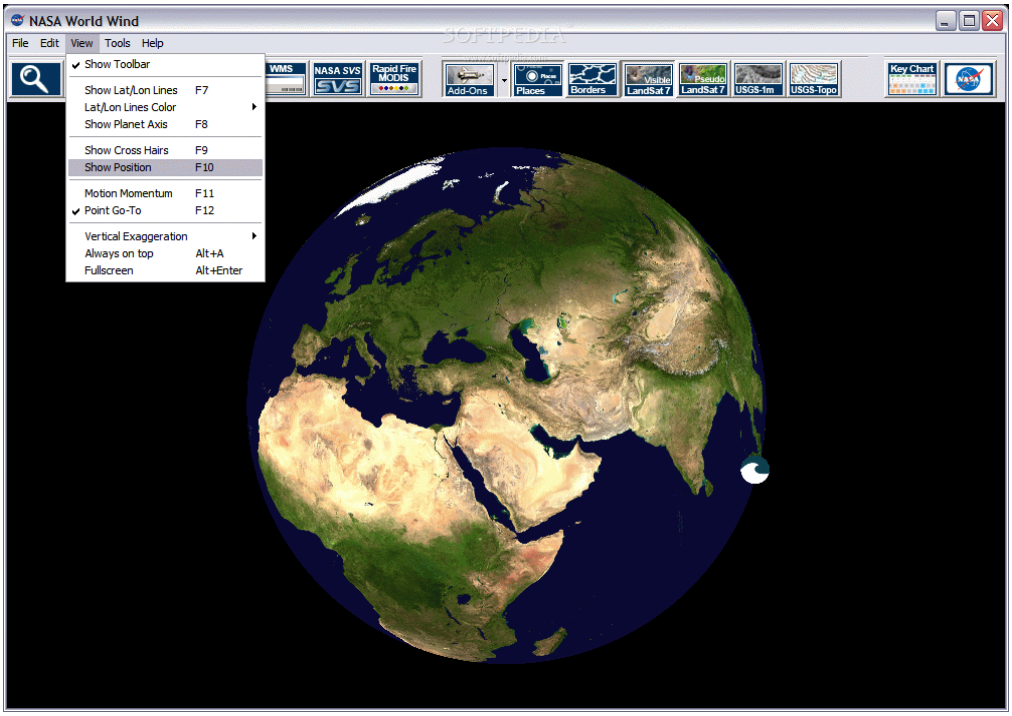
Source License: LGPL

2.3.2 Applications

The applications in the .Net world are not currently tightly bound to the work that has been done at the library level. MapWindow makes some use of libraries from the C tribe, but not of the native C# libraries.

2.3.2.1 WorldWind

WorldWind is a full 3D world browser, in the style of Google Earth, that uses local 3D DirectX hardware and DEM/imagery from NASA to create a complete zoomable world.



Unlike Google Earth, WorldWind is fully programmable and extendable. You do not need to alter the core source code to customize WorldWind (although, because it is open source, you do have that option), you can extend the system using a plug-in facility to add your code to the application. Many of the features shipping in the default WorldWind install are themselves plugins (the Earthquake viewer, for example).

WorldWind does not make use of other members of the .Net family. NASA has since moved most development effort to a Java version of WorldWind (funded by the US Department of Energy), but the .Net version maintains the only mature usable WorldWind currently available.

Maintainer: NASA

Web Site: http://worldwind.arc.nasa.gov/

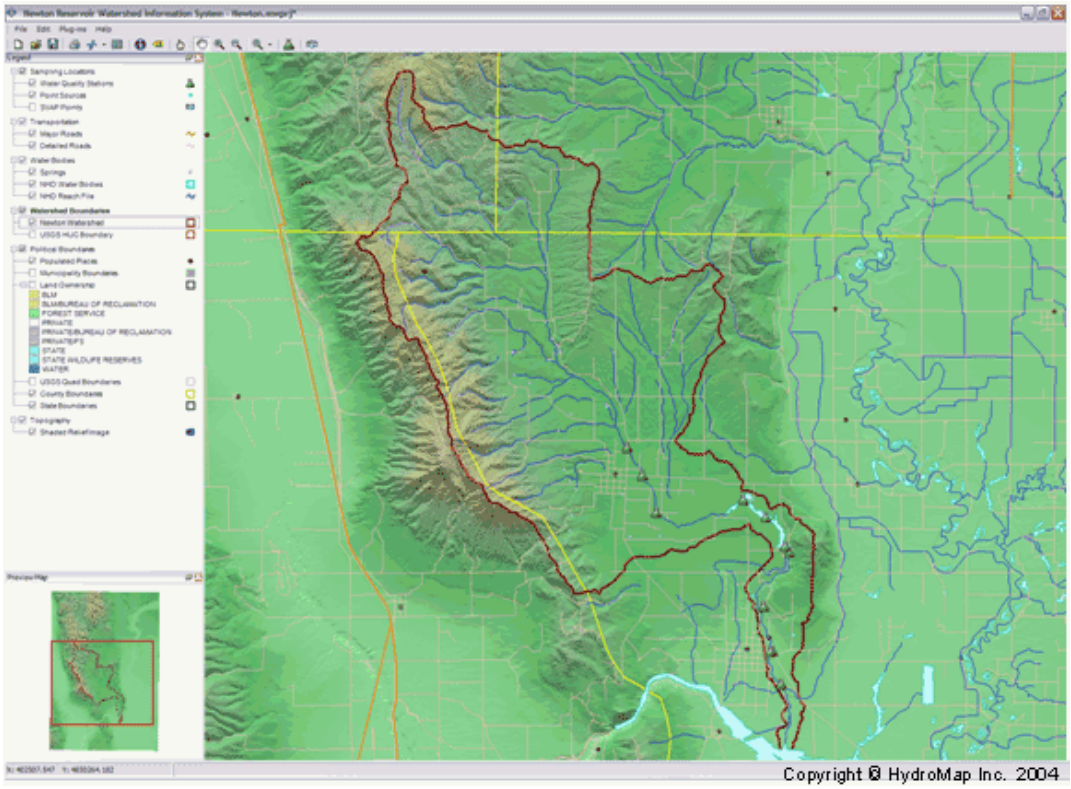
Implementation Language: C#

Source License: NASA Open Source Agreement (OSI Approved)

2.3.2.2 MapWindow

MapWindow is a desktop application written around an OCX control that allows developers to build new applications. It reads and writes Shape files, and can read any format supported by the GDAL/OGR libraries.

As a GIS data viewer, MapWindow can do all the expected operations, styling, thematic views, raster and vector overlays. As an analytical tool, MapWindow provides scripting via the OCX control and standard Microsoft scripting technologies, like VB, or direct access via C#.



MapWindow is used to host the Environmental Protection Agency's "BASINS" model, a model that was formerly hosted on a proprietary data viewer/editor. BASINS is the standard model used for watershed impact assessment, so MapWindow sees lots of use in the real world in watershed impact planning.

Maintainer: Idaho State University (Dan Ames)

Web Site: http://www.mapwindow.com/

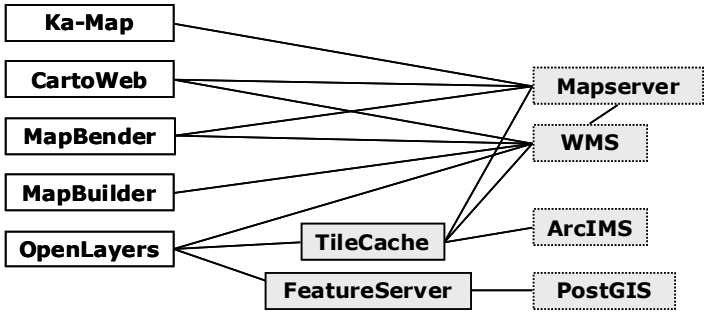
Implementation Language: C#

Source License: Mozilla Public License

2.4 Survey of 'Web' Projects

There are so many web projects, that it is not possible to discuss them all. Instead, we will focus on a few of the larger and more widely maintained web projects, that have spread beyond their initial development stage into a stage of wider community use and maintenance.

The web projects can be grouped into two loose categories: the "toolkits" which are more modular and easily integrated into a custom application; the "frameworks", which can be deployed and run "off-the-shelf" quite capably, but are more suitable for customization than integration.



The dependencies between the web projects and each other are not library dependencies, as in the other charts in this document. Rather, they are potential paths for interoperation. Ka-Map can use Mapserver as a rendering engine. OpenLayers can use Mapserver and WMS servers as rendering engines, and also access them via TileCache. Mapserver can itself ingest maps from WMS engines. And so on.

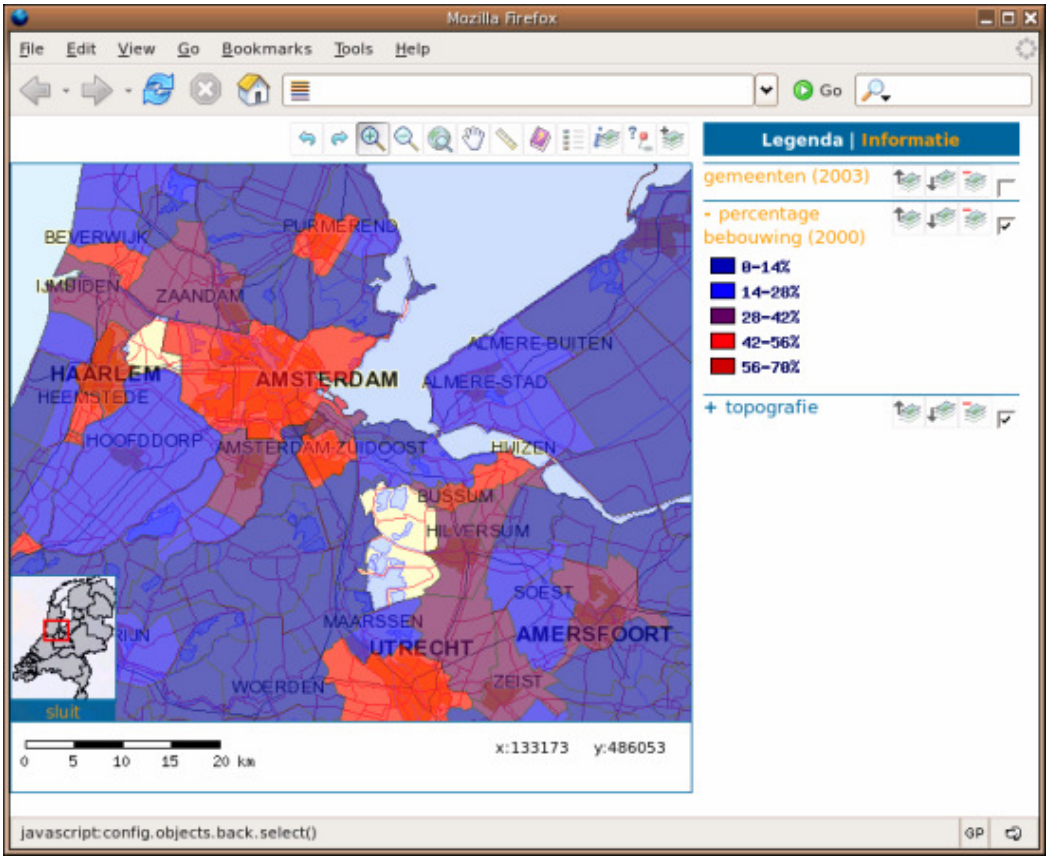
2.4.1 Toolkits

2.4.1.1 MapBuilder

MapBuilder is focused on providing client tools for displaying and editing OpenGIS web services. In particular, pure client side interfaces based on JavaScript. MapBuilder was originally conceived to render Web Map Context documents as specified by the Open GIS Consortium (OGC) in web pages, however the modular design allows MapBuilder to be extended to handle almost any XML document type.

MapBuilder is used in the Canadian geo-portal, to provide key map information alongside metadata. The metadata server returns an OpenGIS "Web Map Context" document, that MapBuilder converts into a browseable map on the client side.

In general, MapBuilder is used for bringing OGC services to the web in a lightweight framework for developers.



Maintainer: MapBuilder Team

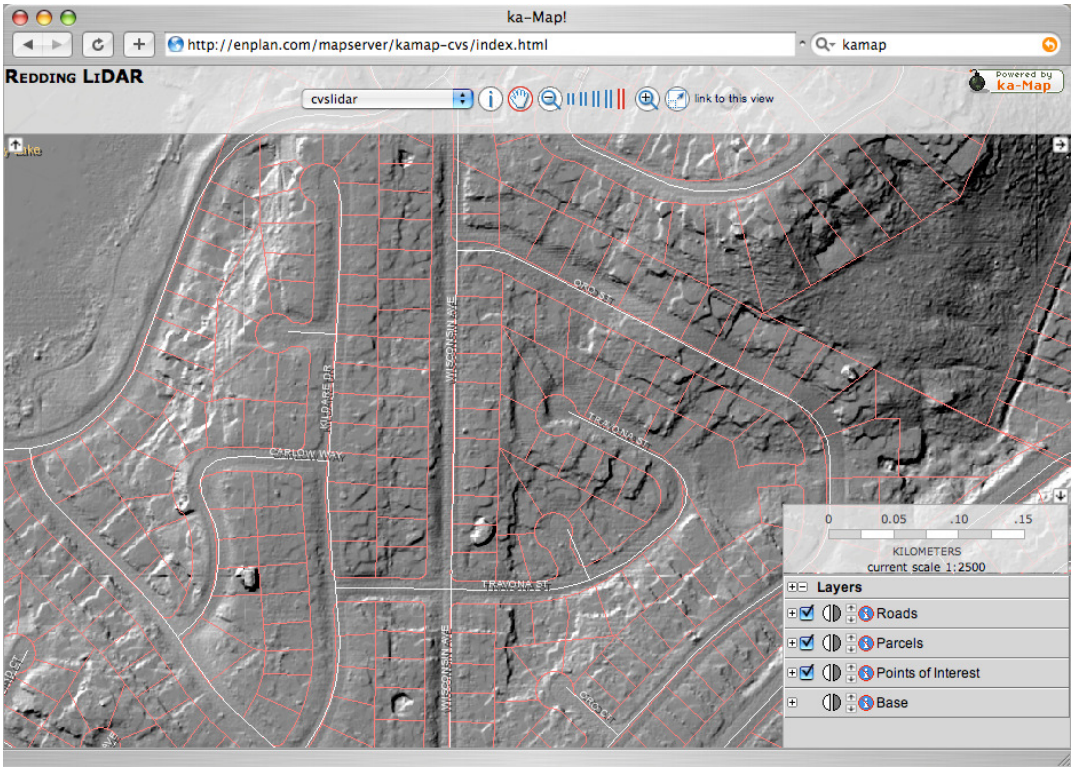
Web Site: http://mapbuilder.sourceforge.net

Implementation Language: JavaScript / DHTML

Source License: LGPL

2.4.1.2 ka-Map!

ka-Map ("ka" as in ka-boom!) is an open source project that is aimed at providing a JavaScript API for developing highly interactive web-mapping interfaces using features available in modern web browsers. In particular, it uses a tile-based map drawing system similar to that used in the popular Google Maps web interface. The result is a particularly smooth and high performance web interface experience.



The most salient feature of the application (the progressive tile loading) is not apparent in a screen shot, but note other niceties such as the alpha-blended control windows and scale bar.

Ka-Map includes both the client-side Javascript web component, and a tightly bound server-side component for tile rendering and caching written in PHP. Because the server-side component depends on Mapscript, it can sometimes be tricky to deploy on non-Windows platforms (anywhere you are compiling Mapscript by hand).

Maintainer: DM Solutions

Web Site: http://ka-map.maptools.org/

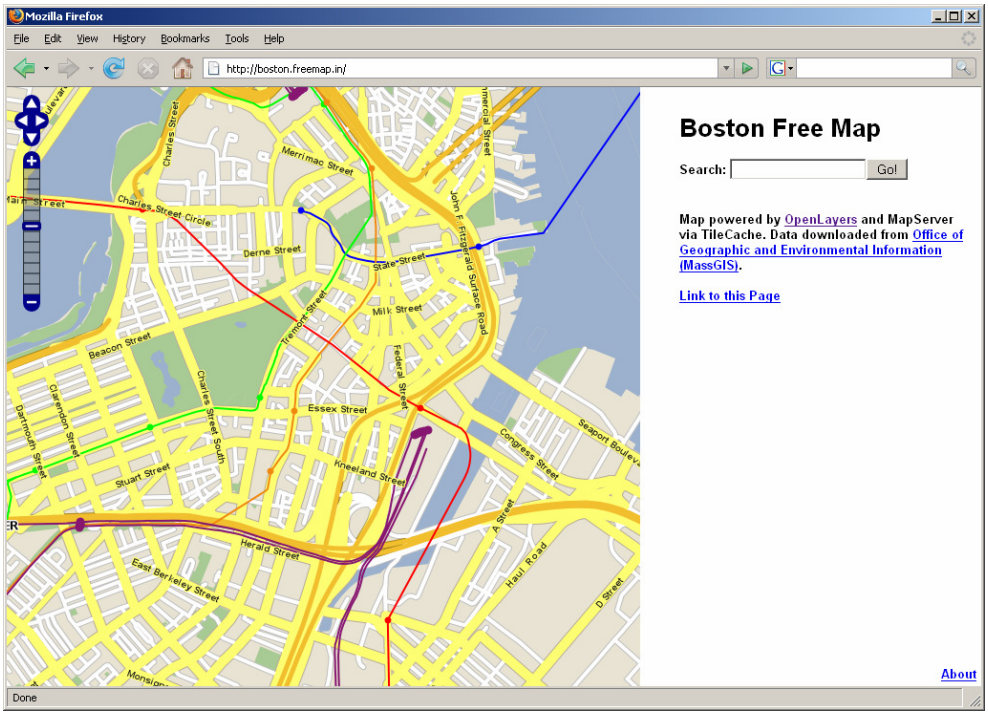
Implementation Language: JavaScript / DHTML / PHPMapscript

Source License: BSD

2.4.1.3 OpenLayers

OpenLayers provides a pure-Javascript, smooth-scrolling client-side library for embedding live maps into web pages. The OpenLayers team made a decision early on to keep their project 100% client-side and not build and specific dependencies on server-side scripting into the application.

The result is an application that is very easy to deploy: just add a Javascript <script> tag reference to a copy of the OpenLayers .js file, and instantly get a map embedded in your web page. The OpenLayers implementation is a tiled map, with continuous panning and discrete zoom levels, like Google Maps.



OpenLayers started as a simple viewer, but with support for multiple sources of map tiles from existing internet services: Microsoft Virtual Earth, Google Maps, Yahoo Maps, Worldwind, WMS services, Terraserver, and so on. It has since added some standard hooks for digitizing and editing vector features on top of the tiled map layer. OpenLayers has a close tie to the other projects from Metacarta Labs, TileCache in particular (for speeding access to dynamic map services like WMS) and FeatureServer (for standard access to database and other backends).

Maintainer: Metacarta Labs

Web Site: http://www.openlayers.org/

Implementation Language: JavaScript

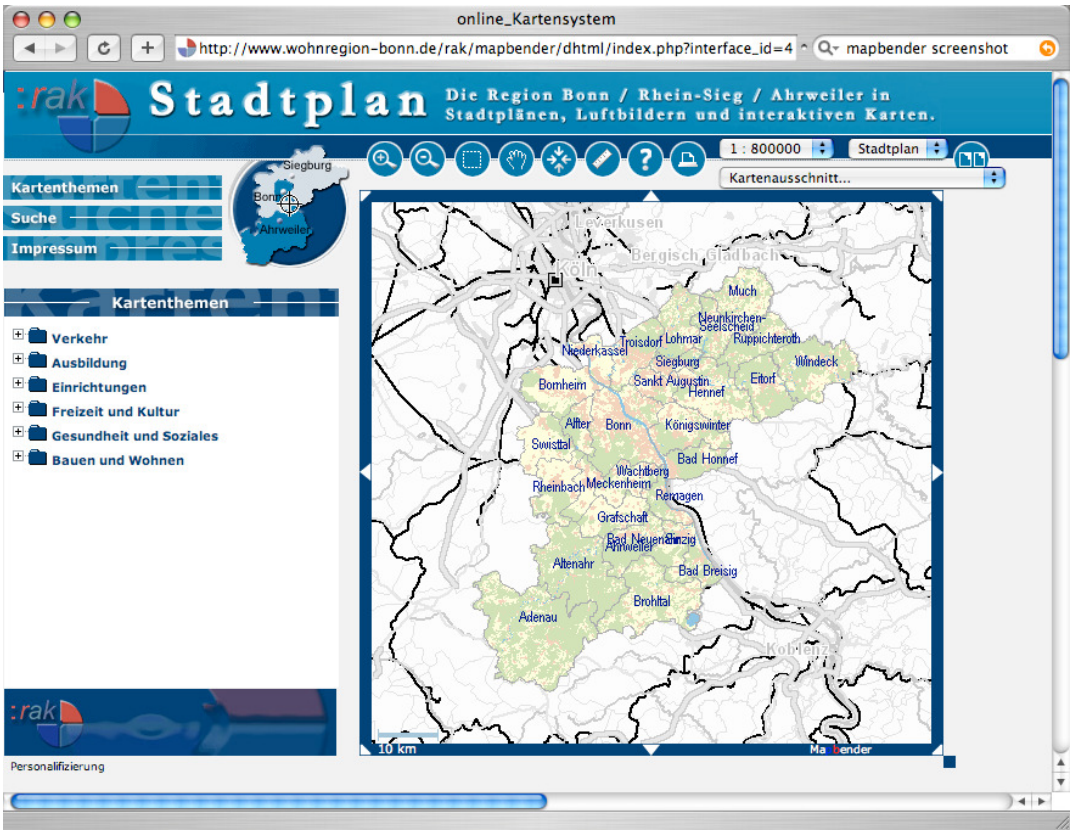
Source License: BSD

2.4.2 Frameworks

2.4.2.1 Mapbender

Mapbender is a project mainly carried out in Germany and used in the German spatial data infrastructure for a number of web mapping sites.

The Mapbender Client Suite is a framework for managing spatial data services. It provides interfaces for displaying, navigating and querying OGC WMS compliant map services. The Mapbender framework contains interfaces for user and group administration, and accessing maps rendered by OGC Web Map Services. The next revision of the software will include support for WFS functionality and catalog services.



Because the primary development and user community is in Germany, a portion of the mailing list traffic is in German, though most developers and users speak English as well.

Maintainer: WhereGroup (info@wheregroup.de)

Web Site: http://www.mapbender.org

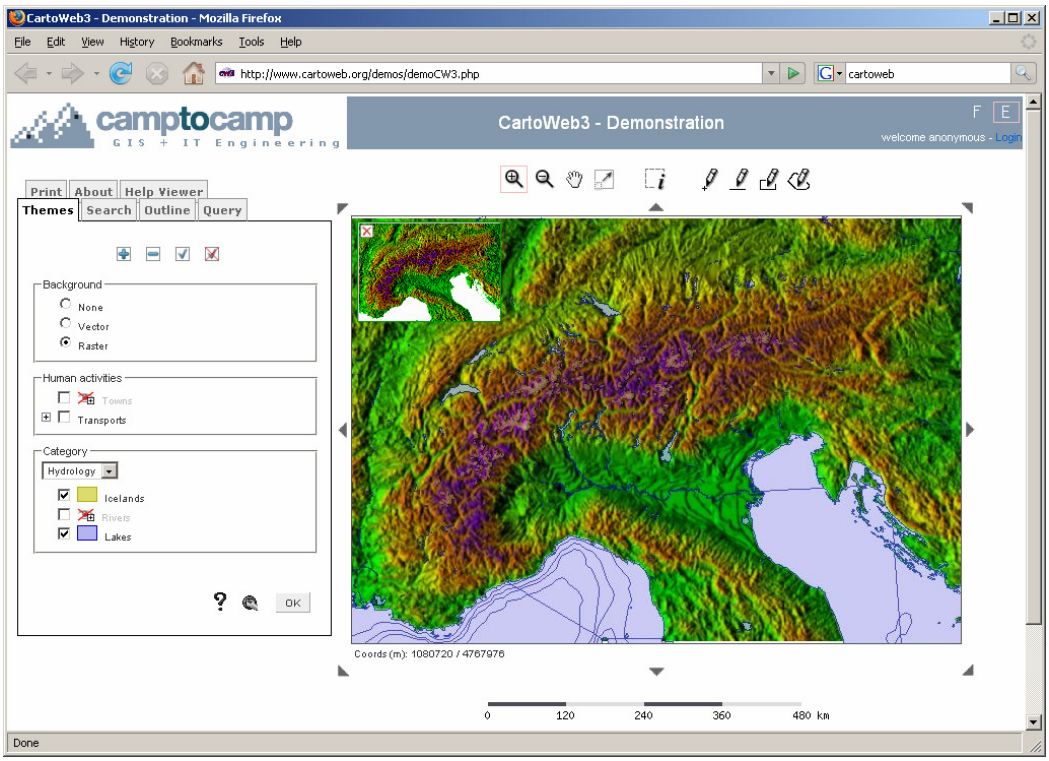
Implementation Language: JavaScript / DHTML / PHP

Source License: GPL

2.4.2.2 Cartoweb

CartoWeb is a comprehensive and ready-to-use Web-GIS (Geographical Information System) as well as a convenient framework for building advanced and customized applications. Developed by Camptocamp SA, it is based on the UMN MapServer engine and is released under the GNU General Public License (GPL).

Written using innovative language PHP5, CartoWeb is highly modular and customizable thanks to its object-oriented architecture. It runs evenly on Windows or Unix-like platforms and shows its real power when associated to PostgreSQL/PostGIS.



Last but not least, CartoWeb may be set up as a SOAP Web Service enabling to have front-end server on one machine and data and map generation on an other.

Maintainer: Camptocamp (info@camptocamp.com)

Web Site: http://www.cartoweb.org

Implementation Language: JavaScript / DHTML / PHP

Source License: GPL

2.4.3 Servers

Of course, all the framework projects include both server (PHP) and client (Javascript) code, but there are two server projects that are rare in being 100% web oriented but not including any UI components. TileCache and FeatureServer serve very limited functional purposes, but serve them very well.

2.4.3.1 TileCache

TileCache sits in front of map generating software, like WMS services, Mapserver, and ArcIMS, and converts requests from tile-consuming software (like OpenLayers, or Worldwind) into requests for maps.

By caching the results of those map requests, TileCache can dramatically speed up the performance of an OpenLayers or Worldwind front end when access otherwise slow map generating services based on WMS, or ArcIMS.

TileCache also supports the generation of tiles using a "metatile" approach, that converts requests for a large number of tiles into a request for one large map, and then cuts the map up into tiles post-facto. Metatiles are used to provide clean rendering from map generators that are doing labeling or other map composition that is sensitive to the placement of the edges of the map.

Maintainer: Metacarta Labs

Web Site: http://tilecache.org

Implementation Language: Python

Source License: BSD

2.4.3.2 FeatureServer

FeatureServer is still an experimental concept, but is likely to see significant growth in the coming years. Like a WFS, FeatureServer provides read/write access to a variety of backend (files, databases) through a common web services interface.

Unlike a WFS, FeatureServer uses simple HTTP commands to control whether the operations are read or write (using HTTP GET, PUT, POST and DELETE), and supports a number of geometry object serializations (JSON, GML, KML).

OpenLayers has optional hooks to do feature editing and server-side storage using FeatureServer.

Maintainer: Metacarta Labs

Web Site: http://featureserver.org

Implementation Language: Python

Source License: BSD