The State of Open Source GIS

Prepared By: Paul Ramsey, Director
Refractions Research Inc.
209 – 560 Johnson Street
Victoria, BC, V8W-3C6
pramsey@refractions.net
Phone: (250) 885-0632
Fax: (250) 383-2140

Last Revised: May 30, 2004

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.

However, 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 licensing 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.

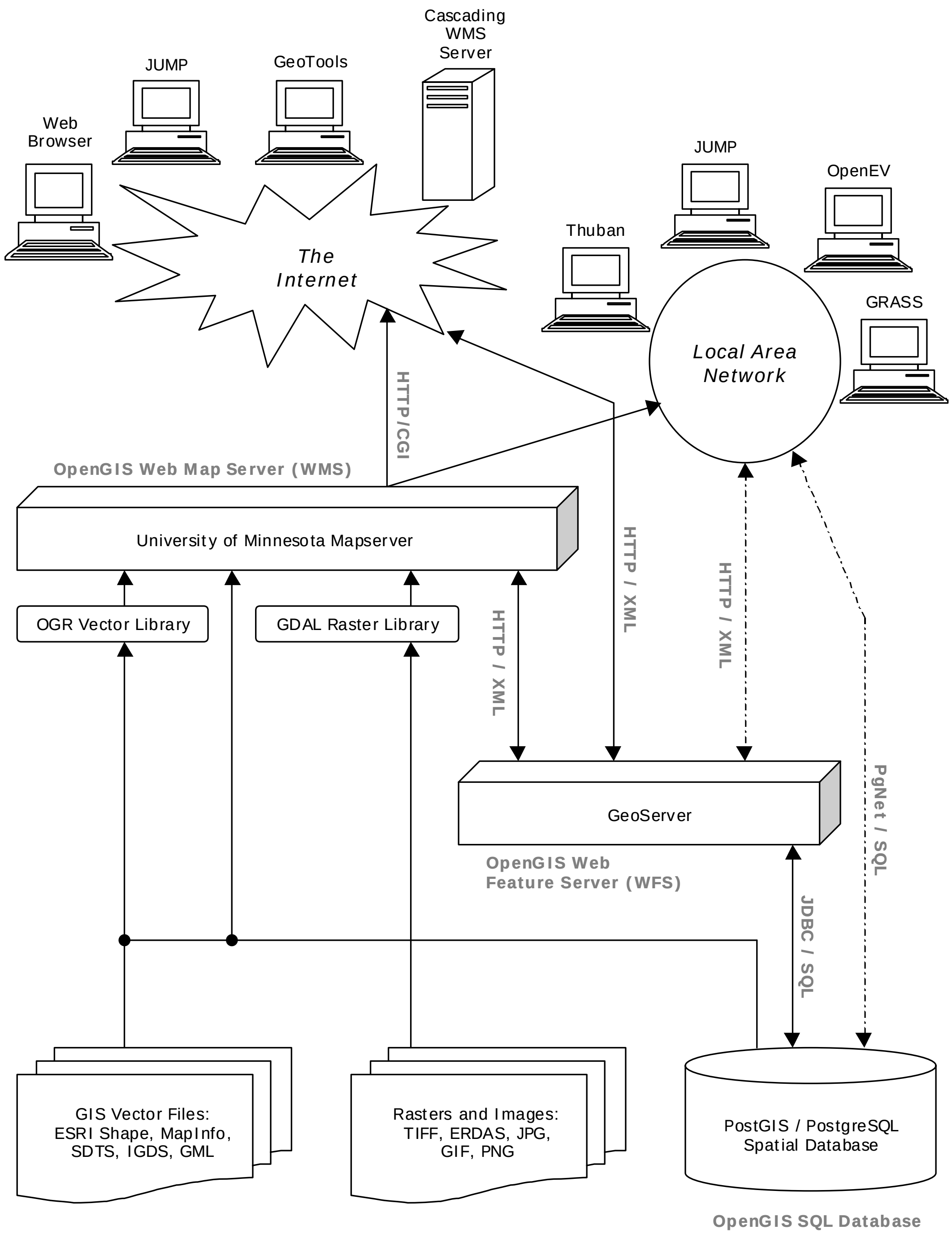
Evaluations of OSS projects should ask:

- **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 two largely independent development tribes. Within each tribe, developers cross-pollinate very heavily, contribute to multiple projects, and have high awareness of ongoing developments. The two tribes can be loosely described as:

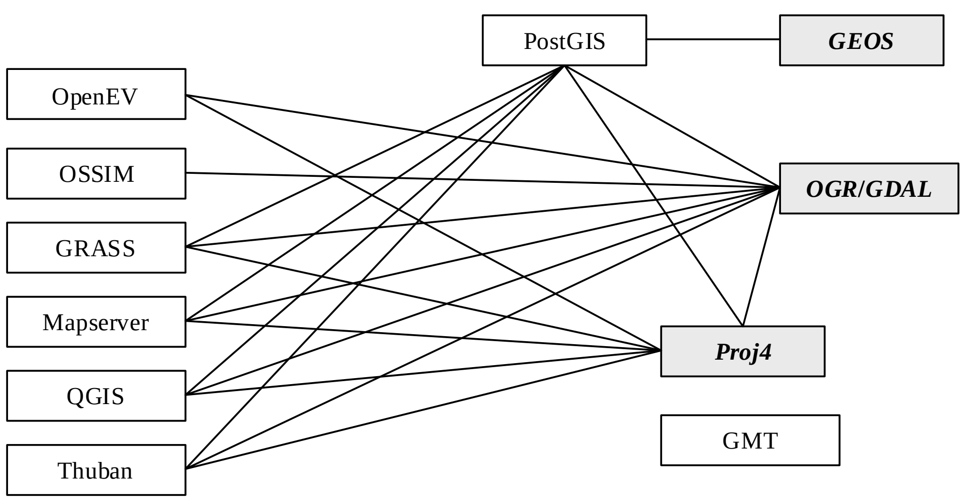
- The ‘C’ tribe, consisting of developers working on UMN Mapserver, GRASS, GDAL/OGR, OSSIM, Proj4, GEOS, PostGIS and OpenEV.
- The ‘Java’ tribe, consisting of developers working on GeoServer, GeoTools, JTS, JUMP/JCS, and DeeGree.

The PostGIS/PostgreSQL project – by virtue of standard database interfaces like libpq (C/C++), ODBC and JDBC (Java) – is used by both 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 protocol.

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.

2.1 Survey of ‘C’ Projects

The ‘C’ projects are, in general, more mature than the Java 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 which 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://remotesensing.org/gdal/

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:

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	--
First Generation USGS DOQ (.doq)	DOQ1	No	Yes	--
New Labelled USGS DOQ (.doq)	DOQ2	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
Arc/Info Binary Grid (.adf)	GIO	Yes	Yes	
GRASS Rasters	GRASS	No	Yes	--
TIFF / GeoTIFF (.tif)	GTiff	Yes	Yes	4GiB
Hierarchical Data Format Release 4 (HDF4)	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	
JPEG2000 (.jp2, .j2k)	JP2KAK	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	
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
ESRI Shapefile	Yes	Yes
GML	Yes	No
IHO S-57 (ENC)	No	Yes
Mapinfo File	Yes	Yes
Microstation DGN	No	No
OGDI Vectors	No	Yes
Oracle Spatial	Yes	Yes
PostgreSQL	Yes	Yes
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 an academic project in geodesy. 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://remotesensing.org/proj/

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 have only a few known failure modes in the spatial operators (geometric functions which produce geometric results).

Important GEOS Methods	
Predicates	**Operators**
Relate(Geom) Touches(Geom) Disjoint(Geom) Intersects(Geom) Contains(Geom) Crosses(Geom) Within(Geom) Overlaps(Geom) IsValid()	Intersection(Geom) Union(Geom) Difference(Geom) Buffer() Distance(Geom) Length() Area()

Maintainer: Refractions Research (info@refractions.net)

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

Implementation Language: C++

Source License: GPL

2.1.2 Applications

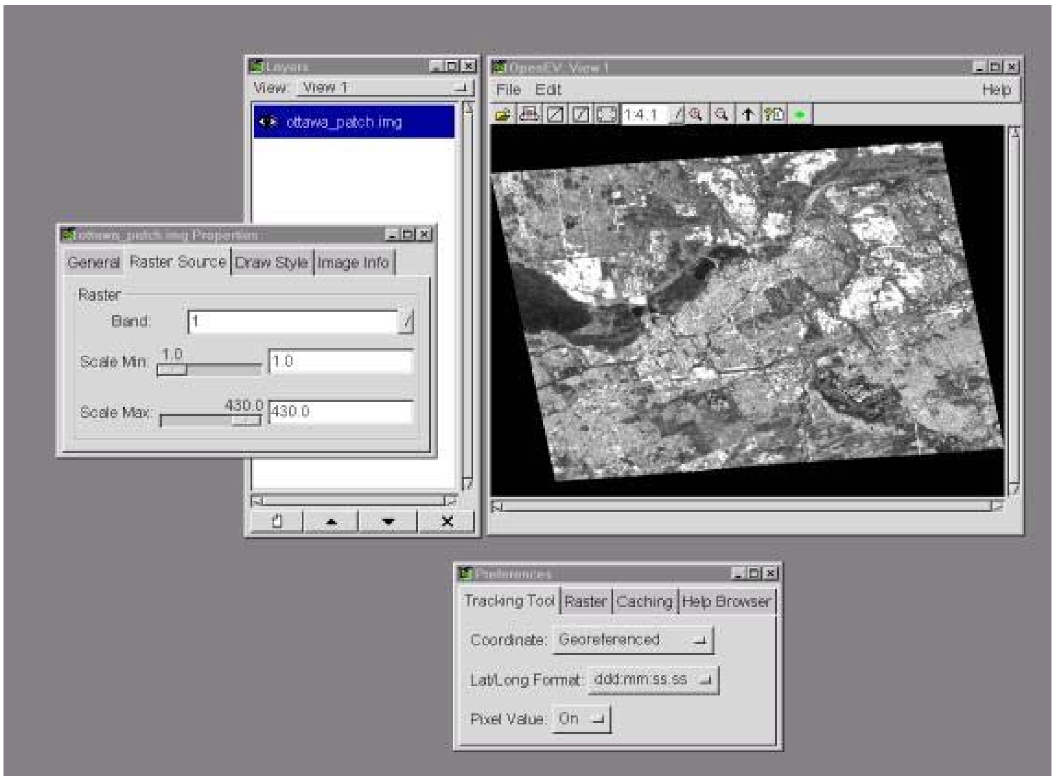
The C family of applications is a mixture of server-side applications and client-side applications, analytical tools and display tools. Most GIS workloads are covered in the application family, with the notable exception of map-making, the most common GIS workload.

> **Note**: The saturated commercial market for cartography tools, the high level of effort to achieve a 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 *OpenEV*

OpenEV is a GIS viewer application, originally designed for a Linux environment but recently ported to work under Windows as well. OpenEV's most interesting design feature is a reliance on OpenGL as a screen rendering language. The reliance on OpenGL means OpenEV can provide very good render performance, but it also restricts the platforms on which OpenEV can be run. OpenEV can quickly view very large image files, and create 3D views of the images in combination with digital elevation files.

OpenEV screen shot:



Maintainer: Atlantis Scientific (http://www.atlantis-scientific.com)

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

Implementation Language: C / Python

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		MapScript Java
JPG/WRL		C API
GIF/WRL		OpenGIS WMS
PNG/WRL		OpenGIS WFS
All GDAL Formats		
All OGR Formats		

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

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

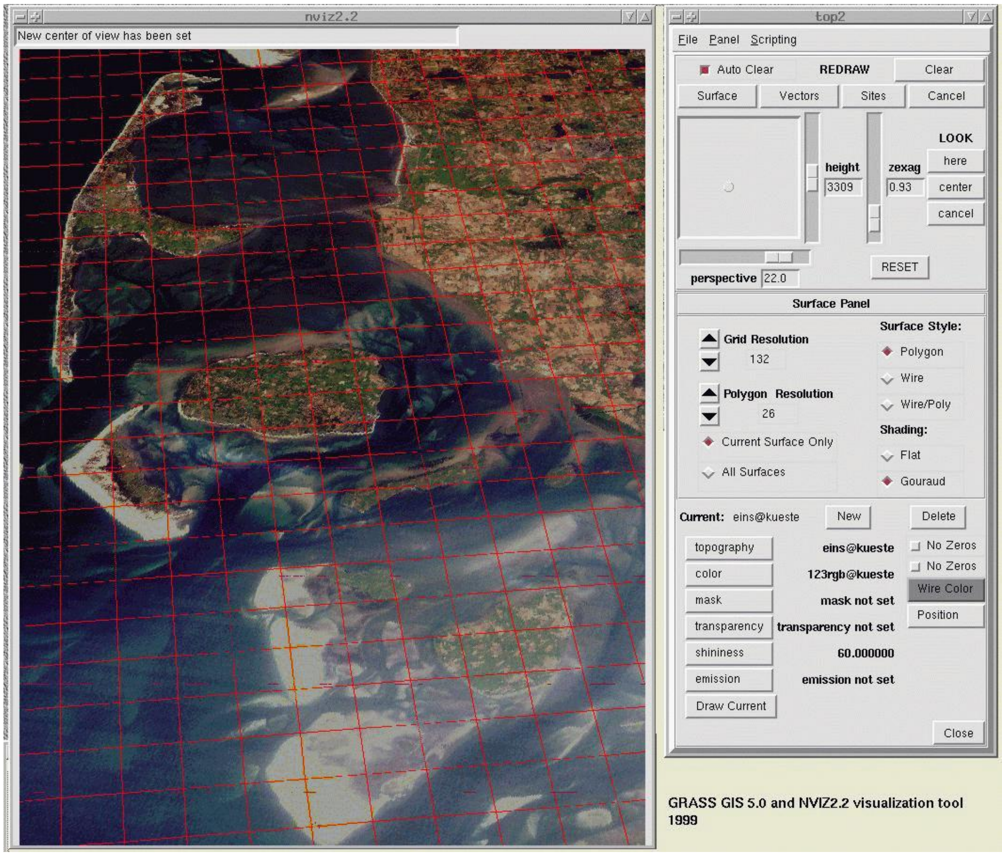
Implementation Language: C

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.

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 GIS 5.0 and NVIZ2.2 visualization tool 1999

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

Maintainer: GRASS Core Team

Web Site: http://grass.baylor.edu//index.html

Implementation Language: C

Source License: GPL

2.1.2.4 OSSIM

OSSIM (Open Source Software Image Map) is a raster manipulation tool chain. OSSIM is primarily developed by ImageLinks (www.imagelinks.com) and is used internally by that company for many image production tasks. ImageLinks also uses OSSIM in their RasterWare 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: Imagelinks Inc

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

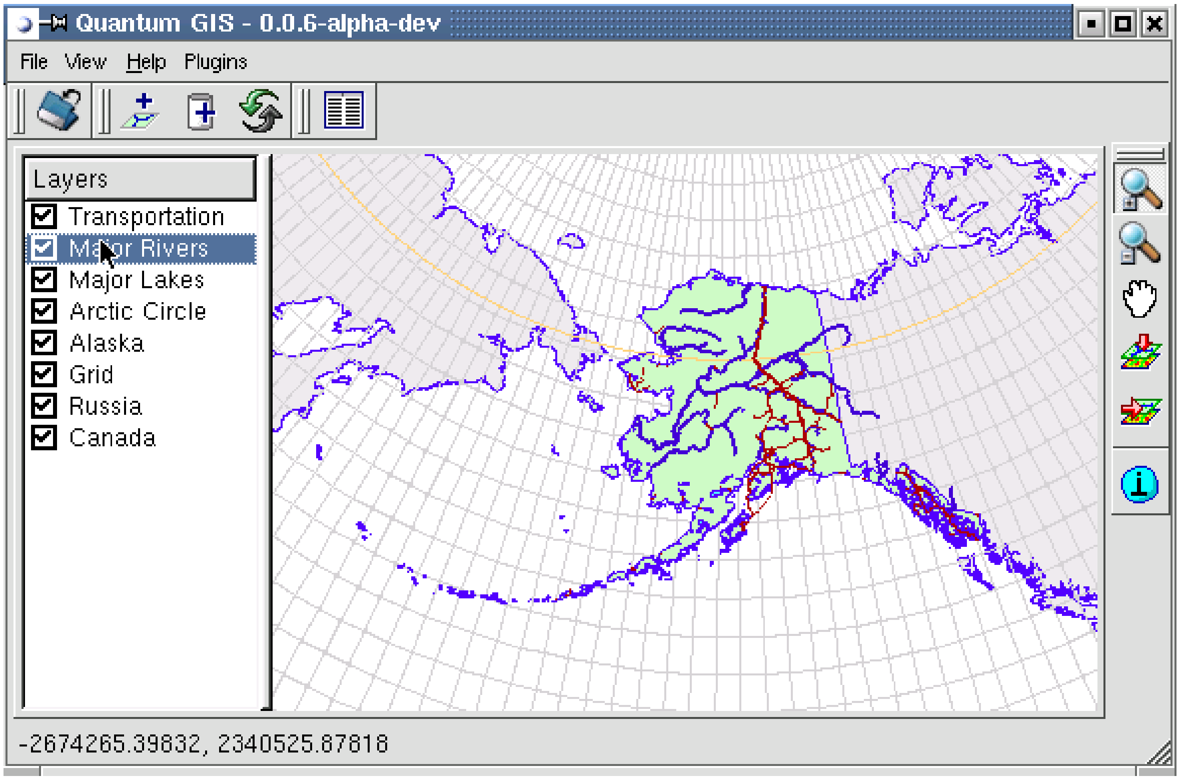
Implementation Language: C++

Source License: GPL

2.1.2.5 QGIS

QGIS is an 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. The developer community is increasing beyond the original founder.

Maintainer: Gary Sherman (gsherman@sourceforge.net)

Web Site: http://qgis.org/

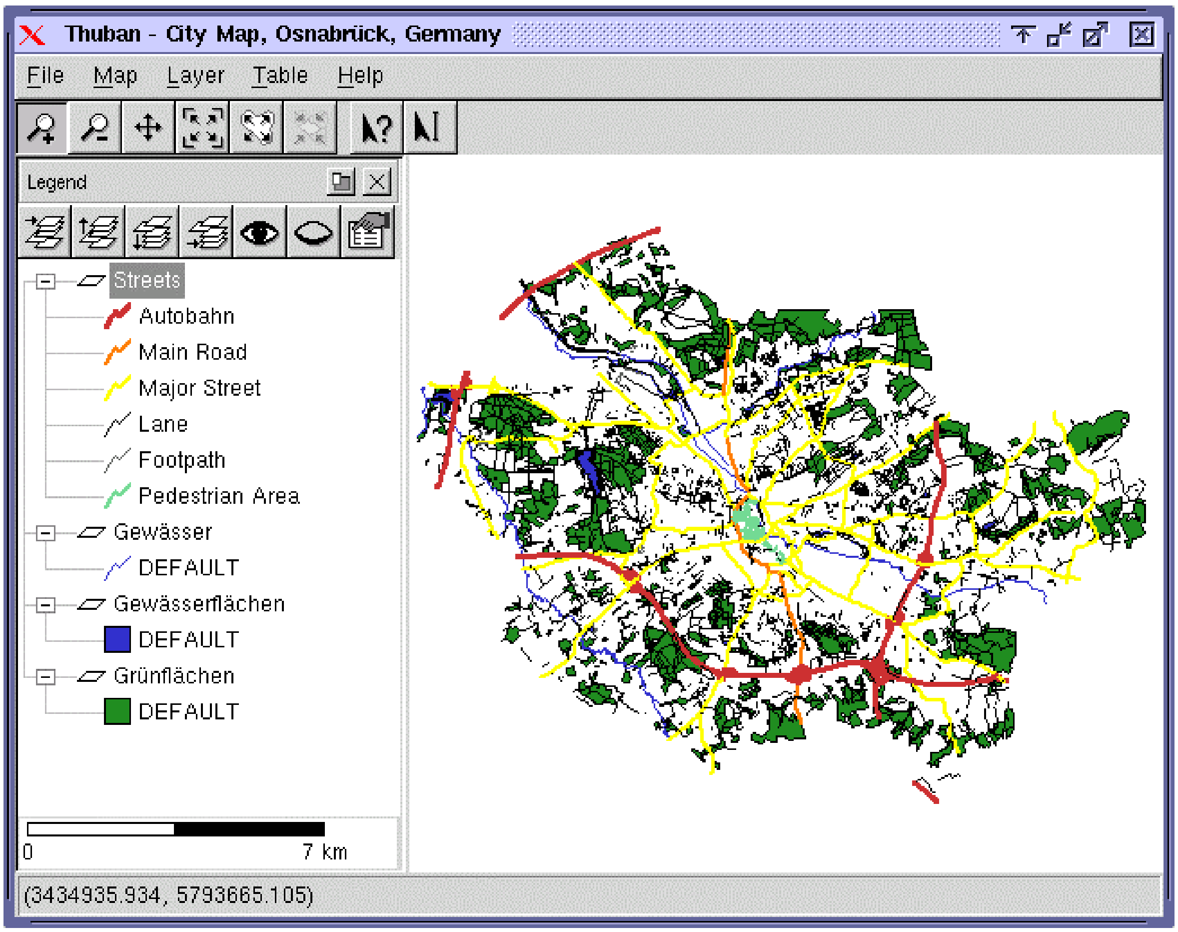
Implementation Language: C++

Source License: GPL

2.1.2.6 Thuban

Thuban is an implementation of a GIS viewer application in Python, using the WxWindows cross platform interface toolkit for the UI. Thuban includes:

- Vector Data Support: Shapefile, PostGIS Layer
- Raster Data Support: GeoTIFF Layer
- Comfortable Map Navigation
- Object Identification and Annotation
- Legend Editor and Classification
- Table Queries and Joins
- Projection Support
- Printing and Vector Export
- API for Add-Ons (Extensions)



Maintainer: Intevation GmbH (info@intevation.net)

Web Site: http://thuban.intevation.org/

Implementation Language: Python

Source License: GPL

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

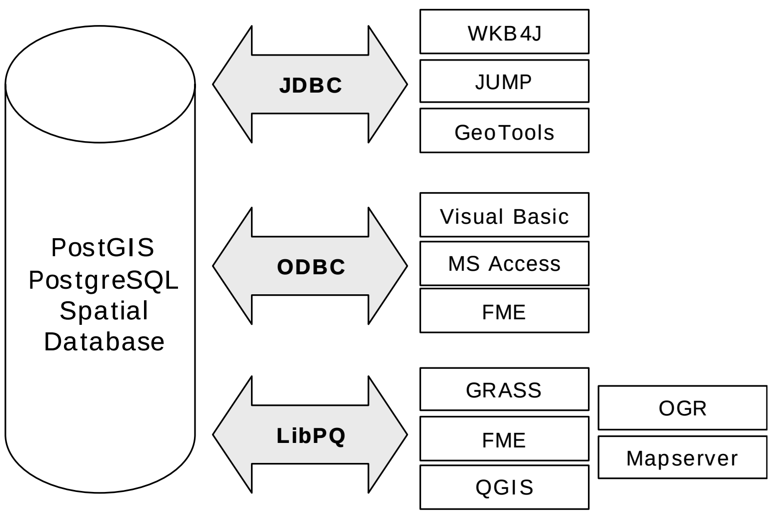
Source License: GPL

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, visualized on the desktop with OpenEV, exported to proprietary formats with OGR.



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

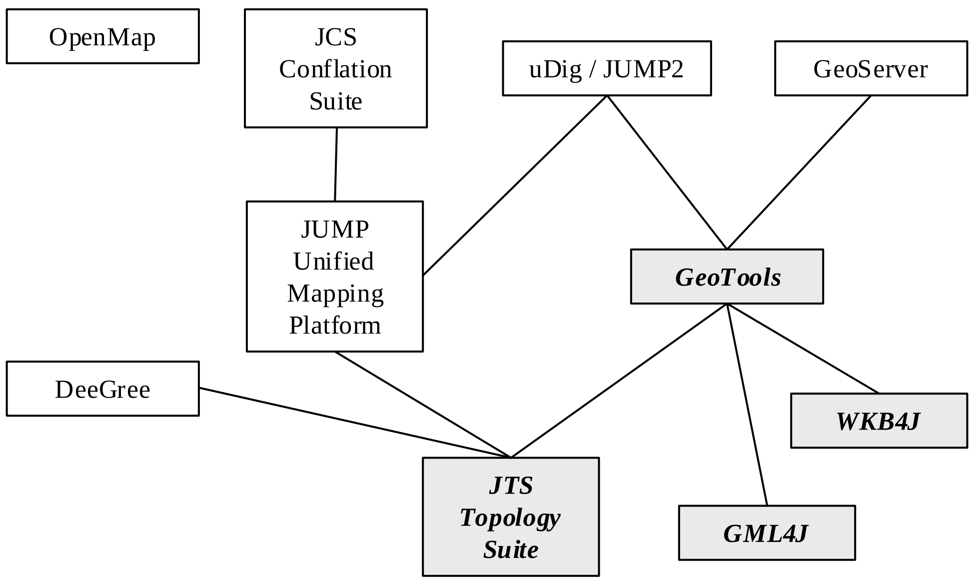
Implementation Language: C

Source License: GPL

2.2 Survey of 'Java' Projects

The "Java" world initially included several independent attempts at "complete unified toolkits" – OpenMap, GeoTools, and deegree. OpenMap continues to be independently developed, but the deegree and GeoTools projects have decided to work together at project convergence. In addition, the new JUMP toolkit project uses many of the same underlying libraries and resources the GeoTools/deegree project does.

As a result, development in the Java world is currently concentrated around projects which use the JTS Topology Suite as the basis for geometry representation.



Side projects, such as GML4J (GML processing) and WKB4J (well-known binary processing) are also used either directly by the projects or by applications which use the toolkit chain.

2.2.1 Shared Libraries

2.2.1.1 GML4J

GML4J is a GML processing library written by Galdos Systems as a test bed for GML technology. It has been used by various of the Java applications for GML processing, but has been largely replaced in favor of event-driven parsers for performance reasons. GML4J uses a DOM (document object model) parsing system, which requires that all the data be held in memory for access purposes. This can result in very large memory footprints for large data sets.

However, GML4J remains the most complete GML processing engine.

Maintainer: Galdos Systems (http://www.galdos.com)

Web Site: http://gml4j.sourceforge.net/

Implementation Language: Java

Source License: Apache

2.2.1.2 WKB4J

WKB4J is a WKB interpretation library developed to proved a high-speed interconnect between Java and WKB-enabled spatial data sources (usually RDBMS). WKB4J provides a “Factory” interface to a WKB data source and can produce a number of different geographic primitive objects – JTS geometries, PostGIS Java geometries, OpenMap geometries.

Maintainer: David Garnier (david.garnier@etudier-online.com)

Web Site: http://wkb4j.etudier-online.com/

Implementation Language: Java

Source License: GPL

2.2.1.3 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 which 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 to 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.

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.

Maintainer: Martin Davis (mbdavis@vividsolutions.com)

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

Implementation Language: Java

Source License: LGPL

JTS development was originally funded by GeoConnections.

2.2.1.4 GeoTools

Geotools is an open source, Java GIS toolkit for developing OpenGIS compliant solutions. It has a modular architecture which 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 support fully 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 1.4.1 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 in as modular a form 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 datasources (e.g Shapefile, MIF/MID, 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 directions of GeoTools is 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 Feature Server specification. 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 GeoServer WFS has been chosen by OpenGIS as a reference implementation for use in the OpenGIS "CITE" interoperability portal. As a reference implementation, GeoServer will be 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 support, GeoServer includes support for the Z39.50 catalog server which is part of the OpenGIS catalog server specification.


```
- <WFS_Capabilities version="1.0.0" xsi:schemaLocation="http://www.opengis.net/wfs
  http://www.refractions.net:8080/geoserver/data/capabilities/wfs/1.0.0/WFS-capabilities.xsd">
  - <Service>
      <Name>My GeoServer WFS</Name>
      <Title>My GeoServer WFS</Title>
    - <Abstract>
        This is a description of your Web Feature Server. The GeoServer is a full transactional Web Feature Server, you may
        wish to limit GeoServer to a Basic service level to prevent modificaiton of your geographic data.
      </Abstract>
      <Keywords>WFS, WMS, GEOSERVER</Keywords>
      <OnlineResource>http://geoserver.sourceforge.net/html/index.php</OnlineResource>
      <Fees>NONE</Fees>
      <AccessConstraints>NONE</AccessConstraints>
    </Service>
```


GeoServer passes all OpenGIS Conformance Tests and is fully compliant with the Web Feature Server 1.0 Specification.

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

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

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 which 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

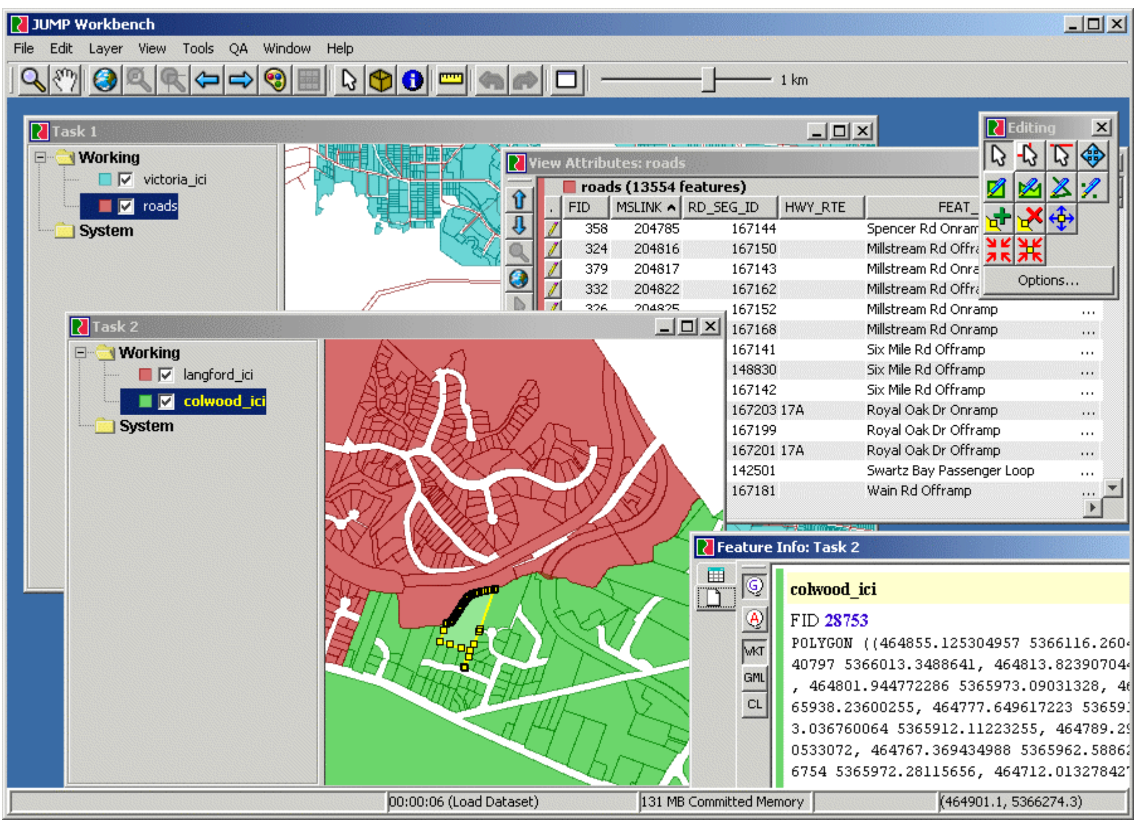
2.2.2.3 JUMP / JCS

JUMP is the “JUMP Unified Mapping Platform”, a visualization and user interface toolkit used by the “JCS Conflation Suite” for solving data integration problems.

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 JavaTM

JUMP supports GML, Shape, and RDBMS data sources.



Maintainer: Martin Davis (mbdavis@vividsolutions.com)

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

Implementation Language: Java

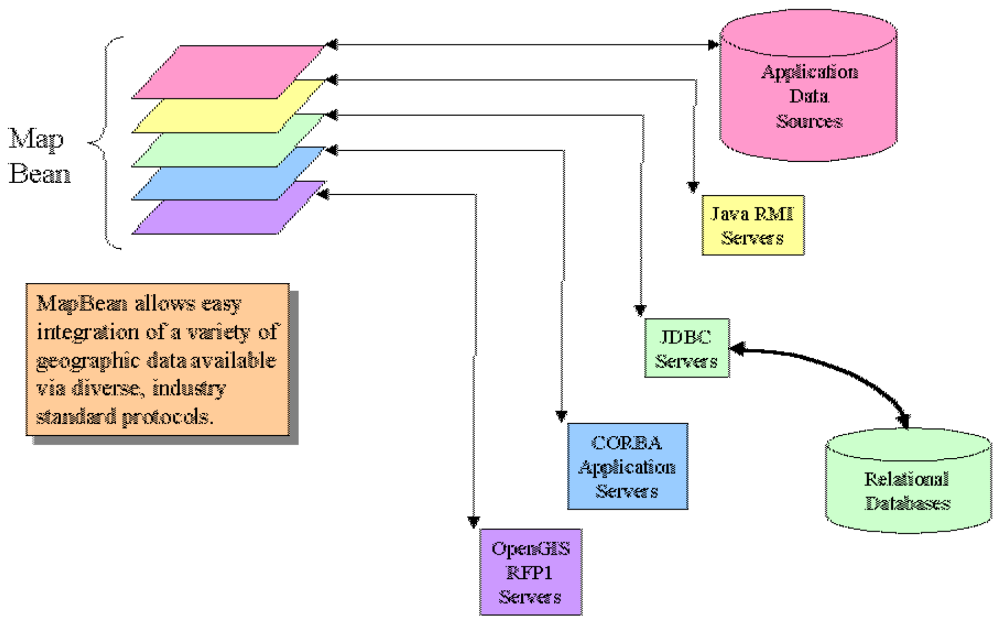
Source License: GPL

2.2.2.4 OpenMap

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

OpenMap is still being actively developed by BBN, and BBN 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

2.2.2.5 uDig / JUMP2

uDig is a project to join the strengths of the GeoTools project (design, data structures, standards) with the strength of the JUMP project (UI, renderer, interactivity) 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 **G**IS", and the goal is to bring internet mapping technologies such as WMS and WFS transparently to ordinary GIS users desktops, creating a worthy successor to the JUMP client environment.

The *uDig* application will have 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 support**, for quick location of available CGDI layers.
- **Printing support**, to allow users to create standard and large format cartography from their desktops using CGDI data sources.
- **Standard GIS file format support**, to allow users to directly open, overlay, and edit local Shape and GeoTIFF files with CGDI 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, Linux and Solaris.
- **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.

At the time of writing, uDig is in the design stage, with a delivery schedule starting in summer of 2004 and running to spring 2005.

Maintainer: Refractions Research (info@refractions.net)

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

Implementation Language: Java

Source License: GPL