The future is open for business: open source tools for the geoscientist

Matt Hall,* ConocoPhillips Canada, provides this primer on open source software and encourages geoscientists to explore further.

Open source software – free software – is everywhere. If you don't like the price of mainstream office productivity suites or image editing software, for example, and you don't like the idea of stealing software, you can easily find open source alternatives: OpenOffice.org and GIMP. Not only are these free, but they are powerful, have responsive support networks, and if you don't like the functionality, you can change them yourself because the all the source code is free, public, and modifiable.

Why would anyone give their source code away? It's a fair question, and it depends upon who you ask. Hardcore activists think software, like speech, should be free, on purely philosophical grounds: 'Free as in freedom,' goes the slogan. Idealists hope science will benefit from the openness and emphasis on reproducibility. Pragmatists hope only to propagate their products, perhaps selling support and training, or plug-ins that extend functionality. Realists recognize it as the only way anyone will ever even try their software.

One thing is certain: open source software is blossoming in geophysics, and it's coming to your workstation. Powerful processing systems, widget libraries, integrated interpretation systems, visualization environments: it's all there. Any skeptics, still unsure whether open source projects can compete with commercial products, or deliver enterprise-grade tools, should note that the Apache HTTP Server, the Firefox web browser, and the Linux operating system are open source projects and are almost certainly in your software portfolio today. And really, the concept is nothing new: software developers have been sharing ideas, one way or another, for years.

Tools for geophysicists

Today there are at least a dozen mature open source tools for geoscience applications, shown in Table 1; most of which are seismic processing toolkits. A handful of them (e.g., FreeUSP, CWP/SU Seismic Un*x, and SEPlib) have been around since the early 1980's and are solid and well-maintained, being used by dozens of practitioners and researchers every day. Others are new, notably Madagascar and OpendTect, both launched in 2003. Having a successful company behind a project helps with rapid development: OpendTect, shown in Figure 1, is maintained by dGB Earth Sciences in the Netherlands.

What is FLOSS?

Free and open source software is something of a minefield of jargon and subtle distinctions. For example, these software packages are variously referred to as public domain software, free software, open source software, free and open source software (FOSS), free/libre/open-source software (FLOSS), and software libre.

A related group of software, referred to as commercial open source software (COSS), is exemplified by free, public MATLAB code, which requires proprietary software to run. Another example of a mixed-source model is dGB's OpendTect project, which has a FLOSS basic framework, but offers enhanced functionality to purchasers of a commercial licence.

Jargon usually arises from a need for nuance, but Richard Stallman, the founder of the GNU Project and a fierce proponent of the open source movement, believes in simplicity. These are his so-called 'freedoms':

- Freedom 0: the freedom to run a program for any purpose.
- Freedom 1: the freedom to examine the source code for a program.

**matt.hall@conocophillips.com*

	Description	Originator	License	Language
Reflection seismic processing packages				
Madagascar	Seismic processing	Sergey Fomel and others	GPL	Fortran 77, C
FreeUSP	Seismic processing	Amoco, BP	Custom	Fortran 77, C
FreeDDS	Seismic processing	Amoco, BP	Custom	Fortran 77, C
PSEIS-OSS	Parallel seismic processing	Randy Selzler	GPL	Fortran 77, C
CWP/SU (Seismic Un*x)	Seismic processing	Stanford, Colorado School of Mines	BSD	C, Fortran 77
SPARC	Seismic processing	ARCO	GPL	Fortran
SEPlib	Seismic processing	Stanford University	BSD	Fortran, C
BotoSeis	Seismic processing	Williams Lima, Brazil	GPL	Java
GeBR	Seismic processing	Various contributors, Brazil	GPL	C
Reflection seismic processing utilities				
GSEGYView	Display SEG-Y files	Vladimir Bashkardin	GPL	Fortran, C
SegyPY	Read and write SEG-Y files	Thomas Mejer Hansen, Univ. of Copenhagen	LGPL	Python
segy-py	Read SEG-Y files	Kurt Schwehr, University of New Hampshire	Custom	Python
Delivery	Bayesian inversion	CSIRO	GPL, BSD	Java
WaveletExtractor	Well-ties	CSIRO	GPL, BSD	Java
GPLib++	Seismic and magnetotelluric processing	Max Moorkamp, Dublin Inst. Adv. Studies	GPL	C++
JavaSeis	Pre-stack seismic I/O	ARCO	CPL	Java
kogeo	Geophysical analysis and visualization	University of Hamburg	GPL	C++
Mines JTK	Signal processing and display	Dave Hale, Colorado School of Mines	CPL	Java
JRG	Modeling and basic processing	John Louie, University of Nevada	Custom	Java
Jive3D	Modeling and tomographic inversion	James Hobro, University of Cambridge	Custom	Fortran
SLIMpy	Processing front end	Felix Herrmann, Univ. of British Columbia	LGPL	Python
Visualization, interpretation & analysis packages				
OpendTect	Geoscience interpretation and visualization	dGB Earth Sciences	GPL or custom	C++
GeoCraft	Geophysical analysis and visualization	ConocoPhillips	MIT	Java
qiWorkBench	Geophysical interpretation and visualization	BHP Billiton	GPL, BSD	Java
Generic Mapping Tools	Map generation and analysis	Lamont-Doherty and University of Hawaii	GPL	C
ParaViewGeo	Geoscience extension of ParaView	Kitware ParaView, MIRARCO	BSD	C++, Python

Table 1 *Some open course software applications for geoscientists working in the oil and gas industry. Please visit the complete 'List of free geophysics software' article in Wikipedia.*

- Freedom 2: the freedom to share a program with others.
- Freedom 3: the freedom to share modified versions of a program with others.

For a more comprehensive description, we can turn to the Open Source Definition from the Open Source Initiative (http://opensource.org). According to the latest version (v1.9), it is not enough just to publish your code. If you choose to protect your work with a licence, and most developers do, there are no less than 10 criteria that your licence must meet to be considered open source:

1. The licence shall not restrict any party from selling or giving away the software with no royalty or other fee for such sale.
2. The program must include source code, and must allow distribution in source code as well as compiled form.
3. The licence must allow modifications and derived works, and must allow them to be distributed.
4. The licence must explicitly permit distribution of software built from modified source code (but perhaps with a different name or version number).
5. The licence must not discriminate against any person or group of persons (notwithstanding applicable laws governing export of some types of technology).
6. The licence must not restrict anyone from making use of the program in a specific field of endeavor (for example being used in a business context).
7. The rights attached to the program must apply to all to whom the program is redistributed without the need for an additional licence.
8. The licence must not be specific to a product, so you can't tie an open source licence to some other collection of software.
9. The licence must not place restrictions on other software that is distributed along with the licenced software.
10. The licence must not depend on any particular technology or style of interface.

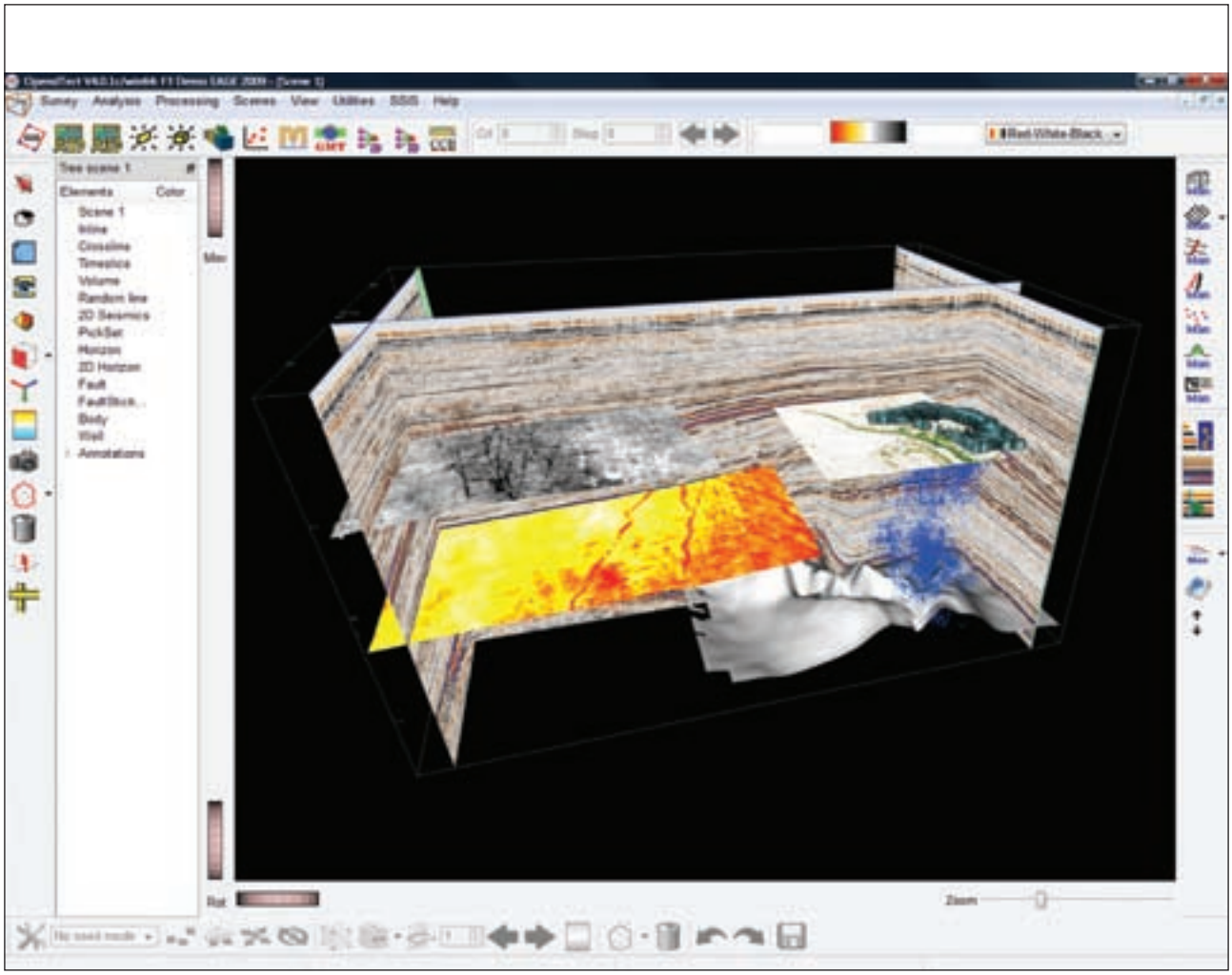

Figure 1 *OpendTect is one of the more mature open-source tools for subsurface interpretation and visualization. It runs on all major operating systems. Image courtesy of dGB Earth Sciences.*

Common licences

Typically, an open source software provider stipulates that, by downloading the software (and, if you choose, its source code or 'codebase'), you agree to the terms and conditions of their licence. There are dozens of flavours of open source software licences (for example, see http://bit.ly/180jm7 and http://bit.ly/GucBM). They can be categorized as permissive or non-permissive (also known as 'copyleft'), with some shades in between.

Popular permissive licences are the Berkeley Software Distribution (BSD) licence, and the Massachusetts Institute of Technology (MIT) licence, sometimes called the X11 licence after the product it first protected. Many *ad hoc* or custom licences are also permissive. The key is that software derived from permissively protected code may be licensed on more restrictive terms than those of the original licence. For example, you can keep the codebase private.

The most important non-permissive licence is the GNU General Public Licence, or GPL. It is regarded by purists as the only true open source licence, but by others, especially those hoping to make a business out of open source software, as a potential show-stopper. The catch with the GPL is that it can force its terms onto any code that incorporates code it protects. This concept is sometimes called reciprocal or 'copyleft', the diametric opposite of copyright; those opposed to the idea characterize it as 'viral'. In other words, if I release software that links to some GPL-protected library, then I may be compelled to open source my entire codebase. If you are not sure that you want this to happen, you should consult a specialist lawyer before using code covered by the GPL.

The Lesser GPL, the Common Public Licence (CPL), and the CPL's new incarnation the Eclipse Public Licence, were all designed to be business-friendly compromises between permissive licences and the GPL. In general, they do this by placing stronger restrictions on modifications of a complete program (you must publish the code) than on merely linking to a library (you don't have to). These licences are sometimes characterized as 'weak copyleft'.

A gross oversimplification may help to explain the situation: there are only two types of licence: GPL and non-GPL. Bottom line: consult a lawyer!

Common languages

The most prevalent languages in open source software are fairly predictable: C, C++, and Java. Among geophysical projects, there are vestiges of Fortran too, but these legacy routines have often been subsumed into C wrappers. One also finds a lot of MATLAB code, especially emanating from academia, but such routines can't really be called open source in the strictest sense, since you require MathWorks MATLAB software to run them (though the open source application Octave does run many programs in this language).

Worth a special mention is Python programming language, a high-level language which many consider well-suited to the needs of science and engineering professionals. It is relatively easy to learn, the code is easy to read, and there are now plenty of open source modules for subsurface scientists. If an ordinary geophysicist wants to spend a little time learning one programming language, it is an excellent choice.

To illustrate its ease of use, here is an example of a complete Python program to load and display a SEG-Y file, using the SegyPY package from Thomas Mejer Hansen at the University of Copenhagen:

```
import segypy
data,header,traceheader = segypy.readSegy
('mydata.sgy')
segypy.wiggle(data,header)
```

It's very short! The first line simply grants the program access to the SegyPY package. The next reads the file. The third plots the data in a new window with SegyPY's *wiggle* command. Exiting the program closes the file and releases the memory automatically. Not everything is that simple, of course, at least until you write your own modules!

The future

There are enough stable and functional applications available today for the keen geophysicist (innovators, early-adopters) to perform most of his or her work in open source software. Users might find issues with installation, data input and output, undocumented bugs, and workflow support. Integrating the tools into existing workflows and data environments will of course be a major challenge too. But if you are prepared to get involved with the development, either as a power user giving feedback on usability, bugs, missing functionality, or perhaps even writing snippets of code, then you should be able to perform almost all your duties as normal (or better).

Neophyte users with business- or time-critical commitments, a need for formal documentation and training, or no desire to be involved in the development process, will find most of today's open source software frustrating and unproductive. But this is just the beginning of a new era in geoscience software development. Five years from now it seems possible, perhaps probable, that several of these tools will have matured, with large user communities and robust workflows. There will likely even be support and training available for some packages (such as are available today for FreeUSP and OpendTect).

Summary

There are dozens of open source (and semi-open source) software applications in our field today. It is already possible to do real work in these tools. But the real value is perhaps in the democratization of geophysics code. We are entering an unprecedented era in which experienced programmers and keen interpretation geoscientists alike can home-brew ad hoc code that piggybacks others' programs for mundane or awkward tasks like data input/output, seismic imaging, signal processing, and visualization. This allows rapid prototyping of new ideas for data processing, fresh takes on data visualization, or novel seismic attributes. We should prepare for – and be part of! – a revolution of innovation.

Further reading

To read more about open source development, visit opensource.org. Layton Payne of dGB Earth Sciences wrote an interesting article on open source tools in oil and gas in Hart's *E&P Magazine*, online at http://bit.ly/a7WBoj. Bill Menger wrote on the same subject in his blog, HPC in Oil and Gas, online at http://bit.ly/beAvsC. Those interested in reading about open source software licensing in more detail might wish to read Rosen, L. (2004), *Open Source Licensing: Software Freedom and Intellectual Property Law*, available free (of course) at http://www.rosenlaw.com/oslbook.htm.

Acknowledgments

I am grateful to ConocoPhillips for permission to publish this article. The views expressed here are strictly mine, and do not necessarily reflect those of ConocoPhillips. Duncan Child (Landmark) is largely to blame for my enthusiasm for open source software, and helped me understand software development and licensing. Thanks are also due to Tooney Fink (ConocoPhillips), Bill Menger (Fusion Petroleum Technologies), Randy Selzler (Data-Warp), Jan Stellingwerff (dGB Earth Sciences), for useful discussion and insight. Any errors, of course, are mine.