Z-99 OpendTect, the Open Source seismic interpretation system

AUTHOR
N. HEMSTRA*
Address
dGB Earth Sciences BV, Nijverheidstraat 11-2, 7511 JM Enschede, The Netherlands
E-mail: nanne.hemstra@dgb-group.com

Abstract

In this paper the success story of OpendTect as an Open Source system is described. In a system like this a free base is provided for general tasks while more advanced functionalities are offered in the form of plugins. The easiness of creating such a plugin is shown. Open Source systems require a large degree of flexibility to be able to support current and future developers' requirements. Therefore OpendTect is developed using an Agile software development methodology.

Introduction

OpendTect is an Open Source software (OSS) system that has been designed for seismic analysis, visualization and interpretation (Fig. 1). It enables geologists and geophysicists to process, visualize and interpret multi-volume seismic data using attributes and modern visualization techniques. Because of its open source structure, OpendTect is not only a functional interpretation system, it is also a research and development environment for seismic analysis. OpendTect users can develop their own interpretation tools as plugins to the system.

Main features of the base system are a/o on-the-fly calculation and visualization of both unique and commonly used filters and attributes, movie-style parameter testing and automatic horizon tracking.

The software is currently supported on PC-Linux, Sun-Solaris, SGI-Irix, Mac-OS/X and MS Windows (2000/NT/XP). Heavy processing of large volumes can be carried out in batch mode on multiple machines. Any combination of platforms and machines (single- or multi-processor machines in heterogeneous networks or clusters) can be used for this purpose.

For more advanced work commercial and free plugins are available. Universities and research institutes have free access to these commercial plugins for education and evaluation purposes and to stimulate new ideas.

The Road to Open Source

OpendTect started out as d-Tect, a commercial product developed by dGB that was released mid 2002. After one year of technical and commercial success it was realized that a) market penetration was not keeping pace with the software's potential and b) the internal

development force was too small to implement all ideas and extensions within acceptable time-frames. Continuing with a proprietary development scheme implied that d-Tect would not evolve into a complete seismic interpretation system as originally intended. It was thus decided to make a radical switch to Open Source. Open Source does not mean “throw it over the wall and see what happens”. In the months preceding the actual opening of the system, a major re-development took place to create a plugin architecture, upgrade the documentation (code, end-user and application management) and generate example plugins. Furthermore, the code was cleaned up and all references to proprietary code were removed. Proprietary commercial functionality (Neural Networks and Dip Steering) were re-developed as plugins to the OpendTect base system. The creation of a new website, opendtect.org, ftp server, e-mail addresses and mailing lists were other essential ingredients.

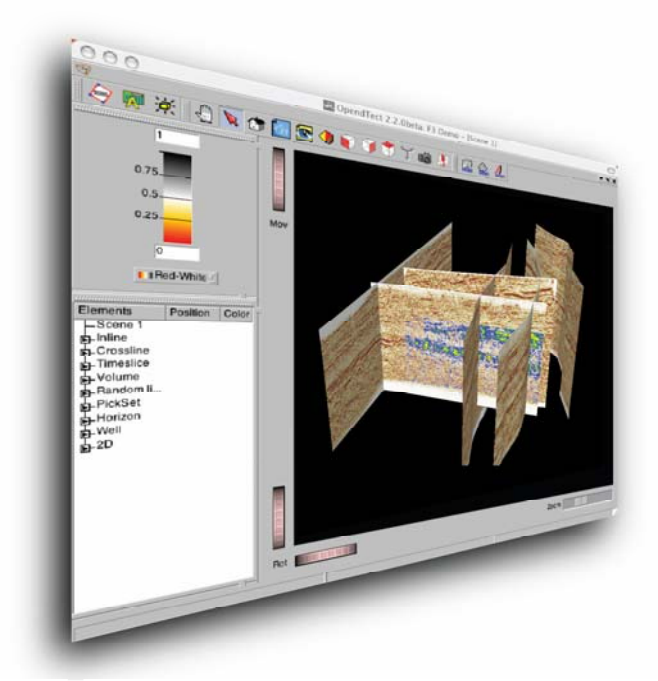

Fig. 1: OpendTect impression

User-driven growth

The overall aim for OpendTect is to support innovative, easy-to-use seismic applications. To safeguard that the system becomes unwieldy by implementing ad-hoc requests from the rapidly growing user community, development possibilities have been grouped into a number of projects that can be sponsored. The way in which OpendTect grows is thus driven by sponsors, hence by the end-users. Development takes place in the open source part of the system as well as in the commercial plugin layer. This is typical. Most development projects are beneficial to both the open source users and the commercial users. Sponsors have to accept that part of their funding is used to the benefit of the (non-paying) open source community. As a rule of thumb general tools and services end up in the open source domain while application specific functionality may yield a new commercial plugin.

Agile Development

It is clear that open source systems require a large degree of flexibility to be able to support current and future developers’ requirements. In the last decade it became clear that Object-

Oriented (OO) technology would provide the tools that enable the creation of complex software in a flexible, reusability-driven environment. These properties of software – flexibility and reusability - are at the core of Agile Software Development (ASD), the methodology that emerged in the new millennium around OO concepts (Fowler, 2000). Although there are Open Source projects not adhering to the principles of ASD, it seems that ASD is far ahead of any other methodology when it comes to countering the impact of changing requirements. This ability to be able to design and integrate new parts quickly is extremely important for OSS. From the start, OpendTect was developed using an Agile software development strategy that was tailored to our geo-scientific needs. This is typical for ASD: the process should be tailored to the domain.

Plugins

An important factor in the success of OSS is the openness for extensions that are not foreseen by the creators. In OpendTect, this is supported by design. Of course it is possible to change the software by modifying existing classes and functions, and adding extensions to the libraries. The advantage is total control. The problem with this approach, however, is that the developer must keep the modified OpendTect sources in sync with new releases. Furthermore, if the developer cannot convince us to adopt the modifications, other OpendTect users may not be able to use the work.

To overcome this problem OpendTect supports a plugin architecture. Plugins make use of all the facilities of OpendTect but are loaded at run-time and can therefore be developed in a completely independent way. One thing a developer cannot do is to use another compiler. OpendTect is built with gcc/g++. Switching to another compiler means that all OpendTect libraries, and supporting libraries like Qt and COIN have to be rebuilt.

In OpendTect, the dynamic library querying and opening is already programmed. A plugin only needs to contain a few standard functions that will be called automatically when the dynamic library is loaded. There are three functions, of which only one is really required. Let's say the name of the plugin is MyMod, these functions will be `GetMyModPluginType`, `GetMyModPluginInfo` and `InitMyModPlugin`. Only `Init...Plugin` is mandatory. The first one, `Get...PluginType` determines whether the plugin can be loaded with OpendTect's auto-load mechanism, and if so, when. 'When' means: before or after the program has created the OpendTect GUI objects (and always after the static objects are initialised). The `Get...PluginInfo` function simply provides info for the users of MyMod plugin.

All functions must be declared 'extern "C', as can be seen in the example plugin "Hello":

```
#include <iostream>
extern "C" const char* InitHelloPlugin( int, char** )
{
    std::cout << "Hello world" << std::endl;
    return 0; // All OK - no error messages
}
```

These 6 lines are enough to constitute a plugin for OpendTect. After compilation the new plugin can be loaded from within OpendTect. At that point in time, the message 'Hello world' should appear on standard output.

To make this a UI - based program we'll need to use functionality from our `uiBase` module. In this case, we use the `uiMSG()` utility:

```
#include "uimsg.h"
extern "C" const char* InituiHelloPlugin( int*, char** )
{
    uiMSG().message( "Hello world" );
    return 0; // All OK - no error messages
}
```

Again 6 lines, but now, after loading, a popup message appears with an OK button (Fig. 2). In this case OpendTect libraries are used and the Makefile should be changed to inform OpendTect's make system about this dependency.

Fig. 2: 'Hello world' popup message

Conclusion

OpendTect is a good example of a successful Open Source model in the geo-scientific world. General tasks belong to a free base, while the more advanced technologies are offered as free or commercial plugins. Thanks to an Agile development strategy, the software is flexible, easy to maintain and can be extended via a strong plugin architecture.

Reference

Fowler, M. 2000. The New Methodology (http://martinfowler.com/articles/newMethodology.html)