EAGE Vienna Workshop 2: Open-Source E&P Software

Introduction

09:00 - 09:05: Dellinger: Why this workshop, and what might it accomplish? (page 2)

1st Morning session: Open source as a business model

09:05 - 09:25: Haase: The business of open standards in the E+P industry (page 3)

Poster session 1 introductions: 5 minutes each

09:30: Hansen: qiWorkbench - an extensible platform for interpretation (page 4)
09:35: Rahon: OpenICarre, an IT solution for E+P software (page 5)
09:40: Hemstra: OpendTect, a seismic interpretation system (page 7)
09:45: Chubak: Integrated geophysical code framework I: kernel (page 11)
09:50: Morozov: Integrated geophysical code framework II: web processing (page 13)
09:55: Gunning: The "delivery" open-source seismic-inversion toolkit (page 14)

10:00 - 10:45: Poster session 1

2nd Morning session: Processing kernels

10:45 - 11:05: Stockwell: Seismic Unix: Past, present, and future (page 16)
11:10 - 11:30: Fomel: Introducing RSF, a computational platform (page 18)
11:35 - 11:55: Dellinger: DDS, a seismic processing architecture (page 19)

12:00 - 12:15: Discussion

12:15 - 13:30: Lunch

Afternoon session: Future speculations

13:30 - 13:50: Selzler: PSEIS, a processing architecture blueprint (page 23)
+Selzler: PSEIS, a blueprint for parallel processing (poster) (page 27)
+Selzler: PSEIS, Meta-data dictionaries (poster) (page 31)

Poster session 2 introductions: 5 minutes each

13:55: Higginbotham: proposed open-source processing system (page 34)
14:00: Hedley: The physical acoustic lab integrated distributed lab system (page 38)
14:05: Bording: Teaching tools for geophysics (page 39)
14:10: Psencik: Software packages from the SW3D consortium (page 40)
14:15: Dai: Converted-wave X-windows tools (page 42)

14:20 - 15:00: Poster session 2

Closing Oral Session

15:00 - 15:20: Hale: Platforms for open-source scientific software (page 46)
15:25 - 15:45: Biondi: Interoperability between open-source packages (page 47)
15:50 - 16:10: Cavazos: Open source E+P software licensing strategies (page 48)

16:15 - End: Panel Discussion

Biondi, Cavazos, Dellinger, Fomel, Glinsky, Hale, Selzler, Stockwell (bios on page 49)

Introduction: Why this workshop, and what might it accomplish?

Joe Dellinger, dellinja@bp.com

Way back in the 1980's, many people had been exposed to Un*x on large computers at school or work. They compared Un*x with what was available to them at home, and knew what they wanted: something that looked just like Un*x but was free, and would run on a small personal computer that they could afford. Several different groups got to work trying to make that dream a reality.

The GNU project replicated an enormous number of Un*x utilities, but never was able to produce a kernel. Linus Torvalds and friends produced a functional kernel, but it was of interest only to a few computer geeks. Scores of others produced all sorts of miscellaneous pieces. None of these comprised a complete Un*x-like system in themselves, but the uncoordinated project as a whole did succeed. Eventually there were enough miscellaneous pieces that a relatively complete Un*x system could be cobbled together. What we now know of as "Linux" was born!

Today in exploration Geophysics, sharing algorithms is difficult. The most common method of widely disseminating a new processing technology is to publish an EAGE or SEG conference abstract. Four pages for the author to describe their method, and then you (and, independently, your counterparts at every other major organization!) get to try to take that and produce a working code. This method of sharing algorithms is enormously frustrating and inefficient, and holds back the rate of progress of the entire field. Too many talented geophysicists are wasting their time doing bad computer science and rebuilding infrastructure already available elsewhere, instead of good research geophysics. The benefits of a widely available open-source package that could be used both for fast-turnaround innovative research and for industrial-strength processing, by academic institutions, contractor companies, and oil companies alike, would be enormous. What is holding us back?

Lots of things, unfortunately, most of which a one-day conference workshop cannot help. However, this workshop can perhaps accomplish two things. The "Linux community" had the advantage of a clear vision of what they were trying to make: Un*x. We have a babel of possibilities instead. This workshop will allow those with good ideas to share them. Seeing these presented together should make it clearer what it is that we could be striving for. This workshop will also allow those with interesting software pieces to show them off. Once we better know what it is that we are trying to create, we may discover that most of the pieces we'll need to make our vision a reality in fact already exist, and just need to be cobbled together. It's happened before!

The "Business" of Open Standards in the E&P Industry

Chris Haase and Michael Glinsky, BHP Billiton Petroleum

Advances in geophysical acquisition and processing technologies combined with the drive for data application transparency and interoperability creates a fertile ground for the creation of industry standards. While petroleum producers have set interoperability demand geophysical processing products with an "open standard" as a top priority, it is not currently shared across the service industry. With the exception of safety, infrastructure and regulatory standards, standards rarely exist within the exploration and production industry – an industry where monopolies have always existed in the production of hydrocarbons and the supply of services to the oil majors. Because business and competition has evolved in such a monopolistic environment, software technology providers are intuitively interested in offering customers a complete, turn-key, integrated system to solve an entire exploration and reservoir imaging problem. Thus, the historical structure of the service industry afforded no incentive for a player to provide compatible modules to a competing imaging system or to allow competing products to be integrated into their geophysical modeling system.

The increased integration of subsurface imaging into project and financial risk management has created high economies of scale among petroleum producers. Also, the uniform application of known imaging and quantitative techniques in the petroleum industry results in a low demand for functional variety among software imaging and interpretation tools. High economies of scale coupled with a low demand for technology variety implies a high likelihood for the emergence of a software standard in geophysical imaging and interpretation tools. As a first step towards realization of this standard, BHP has developed the qiWorkbench, an open architecture for geophysical imaging and processing modules. By enhancing the compatibility and interoperability of processing packages, the qiWorkbench stands to generate value among the user base by (1) increasing the size of the user "network," (2) reducing the risk of technology "lock-in" among suppliers and (3) reducing compatibility problems and (4) increasing innovation and price competition among technology providers.

qiWorkbench – an extensible open-source platform for seismic interpretation

Gil Hansen*, Gilbert.Hansen@bhpbilliton.com
Mike Glinsky, Michael.E.Glinsky@bhpBillition,.com

BHP Billiton Petroleum (Americas) Inc.

Abstract

BHP Billiton Petroleum is developing an extensible open-source geoscience workbench in Java called the *qiWorkbench*. The architecture supports multiple virtual desktops. Each desktop can have multiple interacting applications active at once where an application may be a 2D seismic viewer, a 3D seismic viewer, a Well Log viewer, client-side or server-side IO and job services, an XML editor, core plugins and commercial plugins. The software is freely available at qiworkbench.org.

The workbench comes with 4 core plugins: Amplitude Extraction, Wavelet Derivation, Delivery Lite and Spectral Decomposition. The workbench can be extended by 3rd party plugins. Such plugins can interact via commands with other active applications and build upon core IO and job services. The command set may be extended to accommodate new forms of interaction.

Seismic data and horizon data from multiple vendors is handled by defining a common seismic format (CSF) that components operate on. Transformation services are provided to convert vendor specific data formats to/from CSF. For example, the core Spectral Decomposition plugin operates transparently on Landmark SeisWorks data. The workbench can be extended to handle other vendor data formats by implementing additional transformation services as plugins.

OpenICarre, an innovative open-source IT solution for E&P software

DANIEL RAHON*, DOMINIQUE GUERILLOT, STEPHANE BELLIN (IFP)
daniel.rahon@ifp.fr, dominique.guerillot@ifp.fr, stephane.bellin@ifp.fr

In Exploration and Production, the main way to test innovative methodologies and workflows is through software development. For both research needs and industrial purposes, a specific R&D project has been launched at IFP for several years, for the following reasons:

- Helping multi-disciplinary approaches to gain the best of available data;
- Capitalizing the research work done in this domain;
- Favoring synergies and communication between applications for a more and more extended and flexible workflow;
- Reducing the software development costs and satisfying basic quality requirements;
- Speeding up the transfer of the innovative workflows to the industry.

After so many years, IFP has been developing **ICarre,** its home made software infrastructure and from now on, all new generation software developed by IFP will be based on .this infrastructure.

This E&P common infrastructure supplies a set of basic components such as : data model and persistence, data manager, multiple import/export, 1 to 3D visualization tools, workflow editor and manager, connector for scientific algorithm, etc.

For researchers, the aim is to assist them in the development of software prototypes within the framework of their researches thanks to a management of the purely technical computer aspects. They can so concentrate on the core aspect of their research work to test and validate more easily new algorithms.

The use of a common environment for research prototypes and industrial products will also allow to quickly launch on the market innovative solutions proposed by the researchers at a lower cost.

The philosophy of this platform is to be closely related to published data format such as those of Rescue and OpenSpirit and to be open in order to easily build bridges with other standard tools of the domain.

From a technical point of view ICarre is based itself on *Eclipse* (http://www.eclipse.org) a public domain software platform today widely used in the world of the software development as well as integrated development environment (IDE) or Rich Client Platform (RCP). Other public domain components available on Web are also used by ICarre which thus benefits widely from open source products.

IFP studies today the opportunity of delivering a free version of its platform, namely OpenICarre, to a collaborative open-source exploration and production research community. In that way, well selected COTS (components off the shelf) additionally with innovative components developed by IFP will provide Industry with a comprehensive package for free.

For IFP the interests of this open source version are multiple:

- Test widely the platform;
- Familiarize the community with the concepts and the " look and feel " of its products;
- Favor the exchanges and the interoperability between products IFP and specialized plug-ins which can be finalized by research teams;
- To enrich the platform with a new variety of inputs.

For the research community of the domain and even other areas related to Geosciences, the interest would be to benefit from business oriented development environment, with an industrial quality, which could be augmented and shared with peers.

IFP is also aware that the stake in the public domain of a version of its platform requires an important preparation. Before dashing into this adventure we wish to study the impact that this action could have on the industrial version, define exactly the perimeter of the open source version and especially measure the interest of the research community for this kind of product.

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.

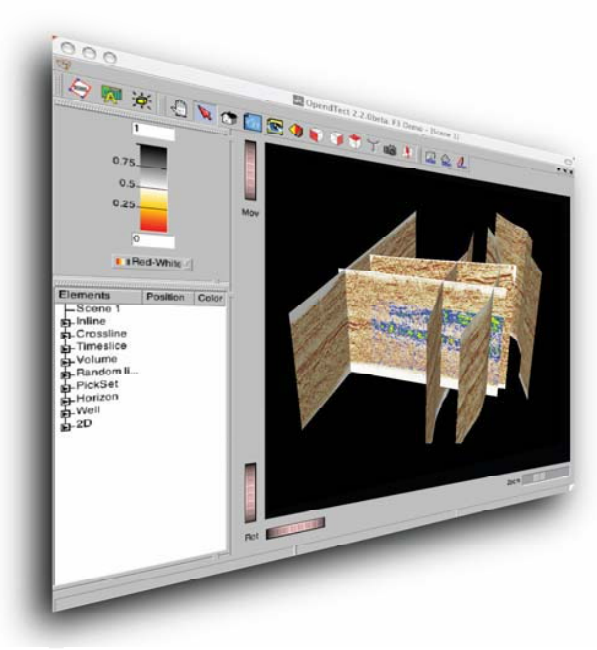


Fig. 1: OpendTect impression

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)

Integrated open-source geophysical code framework: I. Data processing, modelling and visualization

Glenn Chubak and Igor Morozov

University of Saskatchewan, 114 Science Place, Saskatoon SK, S7N 5E2,
Phone:1-306-966-5731
Fax: +1-306-966-8593
Email: gdc178@mail.usask.ca

Open-source software is rapidly gaining acceptance for use in both academic and industry geophysics due to its versatility, large body of code developed, and low cost. In the recent years, the open-source model has received another boost from spreading of Linux operating system. However, most open-source software systems are still limited in their scopes or lack the integration and quality of user interfaces attained by commercial software suites.

The SIA package (http://seisweb.usask.ca/SIA) has been developed as a major effort to provide an open-source code framework to meet the needs of both academic and commercial researchers in several areas of geophysics. SIA includes over 200 dynamically-linked plug-in tools which are closely integrated with a content-agnostic processing monitor and serve a wide range of processing tasks. The current system scope includes reflection, refraction, and to a certain degree earthquake seismology, 2- and 3-D potential field processing and inversion, and graphics. The package consists of three main components operating in parallel and communicating via an object protocol based on the Parallel Virtual Machine (PVM): 1) the user interface, 2) multiple and optionally distributed processing flows, and 3) 3D/2D interactive visualization server.

The graphical interface (Figure 1) is key to making the various features of SIA accessible to users. The interface is an integrated component of SIA and (in principle) allows interaction with running flows. To achieve good performance, portability and modern design it is written in C++ using the Qt libraries from Trolltech. All of the major functions of SIA are available from the GUI, including project management, flow construction, cluster configuration, and software updates. Projects are arranged with the typical Line-Processing flow hierarchy found in commercial seismic processing packages. Flows are constructed by arranging the available tools in a sequence and configuring their parameters.

When a flow is executed, one or several PVM process are spawned and displayed in the monitor window where they can be controlled by the user. The processes can run on a single system and in parallel on peer systems or on a dedicated Beowulf cluster. Each process communicates with the interface through the PVM connection to transfer data, report results or errors, or request input from the user. In addition, various programs, such as the Seismic Un*x, Generic Mapping Tools, or PostScript viewers on multiple compute hosts can be invoked by the jobs.

Recent efforts have focused on integration of seismic interpretation tools through introduction of parallel 2D/3D visualization based on OpenGL. Flows may present their data on any system available through PVM, and multiple displays can operate concurrently on several compute hosts. In this model, processing may occur on a

dedicated cluster but the visualization process can be created on a system with specialized capabilities, such as a stereoscopic GeoWall display. The visualization tool is entirely configurable in the processing flow and allows the end users to create custom applications without any programming. For example, multiple images and buttons could be added to the viewer to create an interactive 2D travel-time picking, ray tracing, and gravity modeling application.

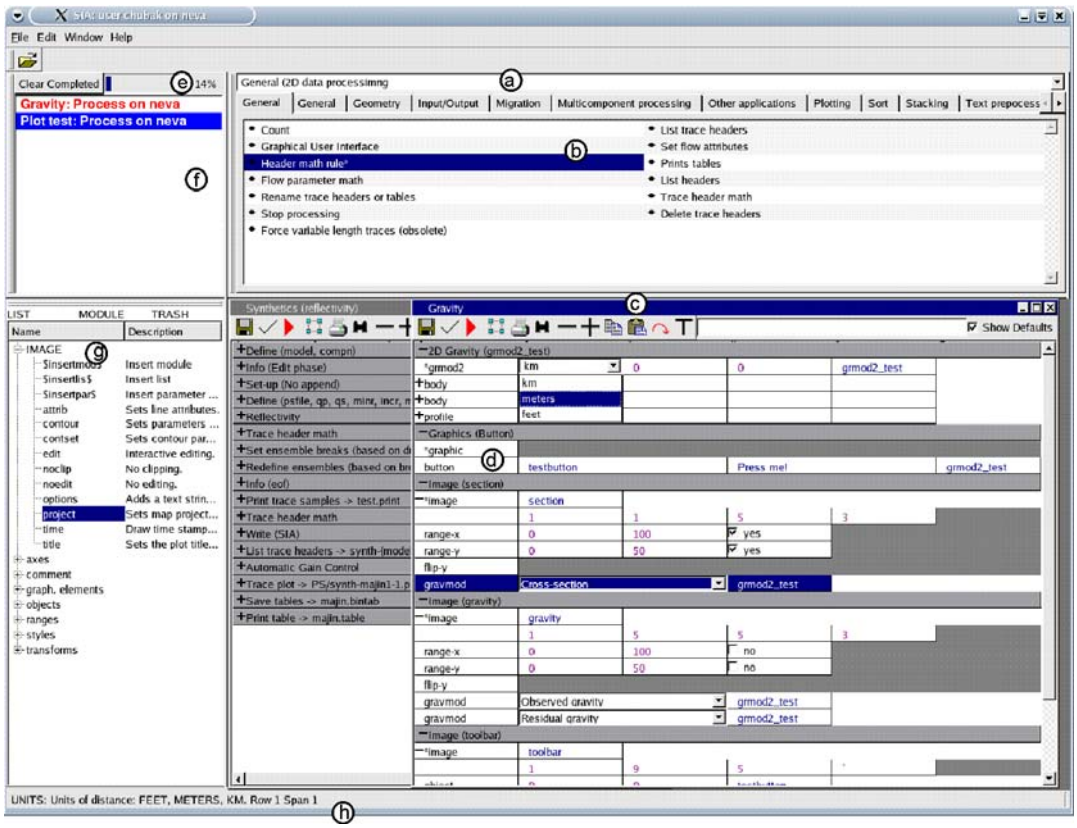


Figure 1. SIA Graphical Interface window including: a) selectable tool packages, b) tool library, c) multiple-job flow editor; d) parameterization of the selected tool; e) job progress bar, f) PVM job monitor, g) tool parameter lists, and h) status bar.

Integrated open-source geophysical code framework: II. Web processing and software distribution

Igor Morozov and Glenn Chubak

University of Saskatchewan, 114 Science Place, Saskatoon SK, S7N 5E2,
Phone:1-306-966-2761
Fax: +1-306-966-8593
Email: igor.morozov@usask.ca

A web service model for data manipulation, analysis, and modeling was implemented, based on a generalized data processing system called SIA. The service is not limited to any specific data type or operation and allows the user to combine ~190 tools of the existing package, with new codes easily includable. Although initially started as a replacement to a commercial reflection seismic processing system, the system has now grown into a generalized code design framework and has been applied in several areas of geophysics, such as reflection/refraction and earthquake seismology, potential fields, and visualization. The system is already fully developed and includes a Graphical User Interface, parallel processing capabilities, on-line documentation, on-line software distribution service and automatic code updates.

The processing service (http://seisweb.usask.ca/SIA/ps.php) allows remote execution of complex processing flows completely designed and controlled by remote clients who are presented with mirror images of the server processing environment. A characteristic feature of the model is its decentralization, with all the servers and clients running identical copies of SIA software. Clients are also able to post their processing flows at the web server, thereby building a knowledge base of processing expertise shared by the community. Flows in this knowledge base are currently represented by a hierarchy of automatically-generated interactive web forms. These flows can be accessed and the resulting data retrieved by either using a web browser or through API calls from within the clients' applications. Server administrator is thus relieved of the need for development of any content-specific data access mechanisms or codes.

Currently, the processing service is utilized to maintain a library of processing examples (http://seisweb.usask.ca/temp/examples) including a number of useful web tools (such as UTM coordinate transformations, calculation of travel times of seismic waves in a global Earth model, and generation of GMT color palettes). Examples of important potential applications of this web service model include building intelligent data queries, processing, and modeling of global seismological data.

In addition to the processing service, a companion code distribution and automatic update service (http://seisweb.usask.ca/SIA/cs.php) was also designed. The service utilizes the SIA system maintenance utilities to synchronize its source code with multiple (user-selectable) software repositories. The program performs automatic checking for the available updates, downloads them from the web, and installs them from source code. Notably, any SIA distribution (used for data processing or code development) on a system providing Apache web services would automatically become a code distribution server.

1⊥

THE "DELIVERY" OPEN-SOURCE SEISMIC-INVERSION TOOLKIT

DR JAMES GUNNING

CSIRO PETROLEUM,
IAN WARK LABORATORY,
BAYVIEW AVE, VICTORIA,
AUSTRALIA 3168

Abstract

We present an overview of the "Delivery" open-source toolkit for (1) stochastic seismic inversion, (2) wavelet extraction, and (3) production of geo-models from "Delivery" inversion models. These three major open-source pieces of software have been produced by CSIRO with support from BHP Billiton in the last 5 years. The first two of these components have been introduced at previous EAGE meetings, the last is a submitted abstract for the present meeting. Here we outline how the whole package fits together and how it interfaces with other open-source projects like SU and the BHP viewer. We believe that significant factors in the success of these developments are the backbone design, the modularity and parellizability of the workflow, and the advantages of the controversial decision to code major components in java.

Brief descriptions of the three components are as follows:

(1) "Delivery" is a model-based Bayesian seismic inversion code. It uses a layer-based prior model with rock physics information takenfrom log analysis as the basic structure that generates reflection seismic data. The model allows for uncertainty in both the fluid type and saturation in reservoir layers: variation in seismic responses due to fluid effects are taken into account via Gassman's equation. Multiple stacks are supported, so the software implicitly performs a full AVO inversion using approximate Zoeppritz equations. Uncertainties and irresolvabilities in the inverted models are captured by the generation of multiple stochastic models from the Bayesian posterior, all of which acceptably match the seismic data, log data, and rough initial picks of the horizons. Post-inversion analysis of the inverted stochastic models then facilitates the answering of commercially useful questions, e.g. the probability of hydrocarbons, the expected reservoir volume and its uncertainty, and the distribution of net sand. The Delivery software is java-based and thus platform independent. Input and output is driven via XML, Seismic Unix1 (SU) and BHP_SU data formats, and the output interfaces naturally with the free BHP viewer. The assumption of independent traces means that the inversion can be massively parallelized, and distributed across cluster computing systems.

(2) The "Wavelet-Extractor" code is a Bayesian wavelet extraction program for deriving wavelets from seismic and well-log information. This code is designed as a companion program to "Delivery" in that the wavelet is extracted using exactly the same forward modelling assumptions as that used in the inversion code. Our program approaches the well-tie problem from a Bayesian view, which is the most natural way to integrate prior knowledge

about the well tie in the form of marker constraints, checkshot data, phase prejudices, and plausible interval velocities. The code can perform simultaneous extractions at multiple (possibly deviated) wells, for multiple offsets in AVO applications (using a linearized Zoeppritz reflectivity), and can estimate additional uncertainty parameters such as time-registration errors for stacks or well-location errors caused by imaging problems. The code produces distribution details for the mis-tie (noise) amplitude, which is critical for inversion studies, and can produce multiple realisations of the extracted wavelets from the Bayesian posterior, showing the uncertainty in the wavelet scaling and extent, the time-to-depth map, and the noise parameters for each stack.

(3) The "Delivery-Massager" module integrates the probabilistic seismic inversion information generated by "Delivery" with reservoir simulation models commonly used in modelling and flow-simulation packages like Petrel. This modelling tool is able to integrate the full range of complex knowledge inferable from probabilistic seismic inversion with auxiliary geological and petrophysical information to produce an integrated model; a process we call "massaging". The major innovative achievement of this code is the synthesis of multi-layer multi-property correlations inferable by the inversion with the transverse correlations induced by geological processes. The massaged model(s) are then directly suitable for uncertainty studies of volumetrics, scenarios explorations, or fluid recovery risking. The Delivery-Massager code is essential for the task of propagating the complex information available from seismic inversion further into reservoir engineering models. Reservoir models honouring the full multi-layer, inter-property constraints available from seismic inversion with well data and geological continuity requirements yield greatly improved estimates of reservoir architecture and uncertainty.

John W. Stockwell, Jr.
Center for Wave Phenomena
Colorado School of Mines
1500 Illinois St
Golden CO 80401-1887 USA

Title: Seismic Un*x: Past, Present, and Future

Abstract
The CWP/SU:Seismic Un*x (SU) package is a free open-source software environment largely developed at the Center for Wave Phenomena (CWP) at the Colorado School of Mines (CSM) in Golden Colorado, USA. What is now known as SU began with a collection of codes called "SY" written in the late 1970s by Stanford University graduate student Einar Kjartansson while he was a student at Jon Claerbout's Stanford Exploration Project (SEP). The SY codes were popularized at Stanford and became part of an early version of the SEPLIB software collection.

With Jon Claerbout's permission Shuki brought SY to CSM in 1985. Shuki Ronen and Jack K. Cohen of CWP conceived of an ambitious goal of turning SY into a general seismic processing software line, which they viewed as being of interest primarily to expert users in the exploration seismic research community. The new package was named SU, meaning both "seismic unix" and "Stanford University" to reflect its beginnings.

From 1986 to the present, the SU package has been expanded and extended, with additions reflecting the software needs and research projects of CWP students and faculty, as well as users outside of CWP. The package grew from Einar's approximately 10 programs, to more than 300 executables. Every member of CWP has contributed in some way to SU. Many users in the world community have provided assistance ranging from bug reports, wish lists, code extensions, and entire suites of new codes for specific applications.

Users of SU include academics, government researchers, commercial software developers, small geophysical contractors, and researchers at larger companies. Individuals interested in seismic processing, near surface geophysics, and ground penetrating radar have found SU useful. Since the first public release of the package (Release 16, in September, 1992) SU has been installed in at least 68 "countries" as identified by Internet Country Code.

The package provides an instant seismic research and processing environment, as well as an environment for the development of new codes. With today's fast PCs and the availability of Linux, Free BSD Unix, and Mac OS X, it is now practical to perform much of 2D seismic processing in a PC environment, allowing practical seismic processing in a classroom environment. The availability of Linux PC clusters has made an extension into parallel processing with MPI a natural extension to 3D applications.

The past 20 years of experience with SU is a source of many valuable lessons regarding open software, technical issues related to software and applications, issues of technical support, as well as legal issues related to licensing and the classification of software as an export.

Introducing RSF, a computational platform for geophysical data processing and reproducible numerical experiments

Sergey Fomel*, University of Texas at Austin, Paul Sava, Colorado School of Mines, and Felix Herrmann, University of British Columbia

This talk is the first public announcement of RSF (from *regularly sampled format*), an open-source software package developed in collaboration by a group of geophysicists, petroleum engineers, and computational scientists. It is also an open invitation for other scientists to join the project as users or developers.

The four main features of RSF are:

1. RSF is a *new* package. It started in 2003 and was developed entirely from scratch. Being a new package, it follows modern software engineering practices such as module encapsulation and test-driven development. A rapid development of a project of this scope (more than 300 main programs and more than 3000 tests) would not be possible without standing on the shoulders of giants and learning from the 30 years of previous experience in open packages such as SEPlib and Seismic Unix. We have borrowed and reimplemented functionality and ideas from these packages.

2. RSF is a *test-driven* package. Test-driven development is not only an agile software programming practice but also a way of bringing scientific foundation to geophysical research that involves numerical experiments. Bringing reproducibility and peer review, the backbone of any real science, to the field of computational geophysics is the main motivation for RSF development. The package consists of two levels: low-level main programs (typically developed in the C programming language and working as data filters) and high-level processing flows (described with the help of the Python programming language) that combine main programs and completely document data processing histories for testing and reproducibility. Experience shows that high-level programming is easily mastered even by beginning students that have no previous programming experience.

3. RSF is an *open-source* package. It is distributed under the standard GPL open-source license, which places no restriction on the usage and modification of the code. Access to modifying the source repository is not controlled by one organization but shared equally among different developers. This enables an open collaboration among different groups spread all over the world, in the true spirit of the open source movement.

4. RSF uses a *simple, flexible, and universal* data format that can handle very large datasets but is not tied specifically to seismic data or data of any other particular kind. This "regularly sampled" format is borrowed from the traditional SEPlib and is also related to the DDS format developed by Amoco and BP. A universal data format allows us to share general-purpose data processing tools with scientists from other disciplines such as petroleum engineers working on large-scale reservoir simulations.

W-2 DDS, A Seismic Processing Architecture

RANDALL L. SELZLER, JERRY EHLERS, *JOSEPH A. DELLINGER
RSelzler@Data-Warp.com Jerry.Ehlers@BP.com dellinja@BP.com

Abstract: The Data Dictionary System (DDS[1]) is a seismic processing architecture. It has proved its value in a *production* environment at a leading High Performance Computing center over the last ten years. It also provides the flexibility needed for advanced *research* in seismic imaging. The software has been ported to more than a dozen hardware platforms and the data sets can be efficiently accessed across any combination of them. One unique feature of DDS is its ability to inter-operate with other processing systems. This is accomplished using a combination of generic data formats, interface emulation and on-the-fly conversion. Inter-operation can protect your investment in existing software, leverage the unique capabilities of other systems, facilitate collaboration with partners and provide an evolutionary path to the future that complements the open source philosophy.

DDS was released by BP America in 2003 under an open source license. The goal of this paper is to reveal useful innovations so that subsequent projects can benefit from DDS.

Introduction: The Stanford Exploration Project[2] was one of the first to provide seismic processing software under an open source license. Equally important is Seismic Un*x[3]. They were conceived in the 70's and 80's within an academic environment and have been used to some extent commercially. FreeUSP[4] is a contemporary package with a commercial origin. It was made available to the public in 2001 (before DDS) by BP America.

Development of DDS began in 1995 at the Amoco Research Center, in Tulsa Oklahoma. Many of its concepts were borrowed from other systems, including SEPlib, USP, SU and Cogniseis DISCO. The infrastructure emphasizes a generic seismic format, interfaces to other processing systems and high-performance disk I/O.

Seismic processing architectures can be divided into two broad categories. Architectures such as DISCO and ProMax use an executive program to control the flow of data and call coroutines to operate on it. Architectures such as SEPlib, SU and FreeUSP rely upon the OS to execute independent processes and transfer data among them. DDS falls into the later category.

DDS has one underlying philosophy: Data handling is fundamental. Get it right and other issues will fall into place. Get it wrong and the system is doomed to mediocrity.

DDS software supports a variety of platforms. It is currently used by BP on workstations (Sun-Solaris, Intel-Linux), servers (SGI-IRIX) and high-performance clusters (Intel-Linux, HP Itanium-Linux, SGI Altix-Linux). BP's HPC center has thousands of compute nodes, many terabytes of memory and petabytes of raid disk storage (managed by 5 systems administrators!). DDS has also been used on Cray, Thinking Machines and Convex super-computers. It provides an Application Program Interface (API) for C and Fortran. Data sets are portable. Files and I/O streams created on one platform are automatically converted as needed when accessed by another platform.

A persistent challenge faced by seismic processing software is efficiency, flexibility, portability and inter-operation with other processing systems. The architecture used for data handling has a major impact on all of these issues. The importance of the first three is obvious and most processing systems solve them to varying degrees. I believe the last one, inter-operation, is very important, but is often neglected. DDS provides a solution to all four issues.

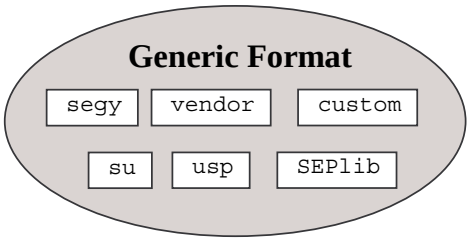


Figure 2: Generic format.

In a perfect world, *everyone* would just use *my* favorite processing system. In the real world, inter-operation with other processing systems has significant value. It can protect your investment in existing software. It allows you to leverage the unique value provided by other systems. It facilitates collaboration with partners in industry and academia. It also provides an evolutionary path to the future that complements open source software.

Data Dictionary: DDS normally keeps binary data and dictionary information in separate files (Figure 1). The importance of this concept is reflected in the name, *Data Dictionary* System. This concept was patterned after SEPlib circa 1994. It kept trace samples in a binary file (typically a cube of numbers), and meta-data in a history file (in plain text). This division of responsibility is simple and flexible.

Multi-dimension: DDS is designed to handle multi-dimension data sets. Each dimension can be described by an extensible set of attributes. This allows many problems to be accurately described in their most natural terms. The dictionary in Figure 1 shows an excerpt for a simple data set with three dimensions. The number of dimensions, their names and nominal sizes are specified in a simple and intuitive syntax. The format of the binary data and its filename are also obvious.

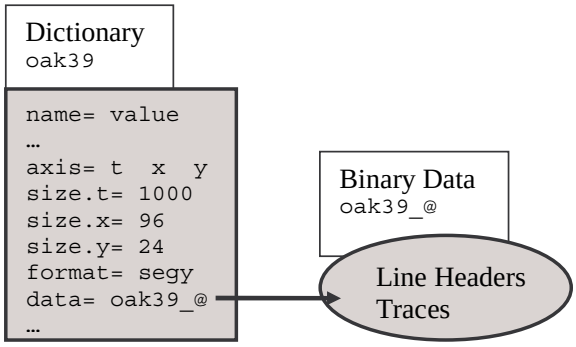


Figure 1: Dictionary and data file for 3D data set.

The axis names and nominal sizes are required by the DDS API. Meaningful names can be chosen for time (`t`), space (`x, y, z`), transforms (`w, kx, ky, kz`) and field geometries (`shot, station, channel, cdp, offset`). In addition to size, most dimensions have attributes for physical coordinates (`origin.x, delta.x, units.x`) or index schemes (`base.x, step.x`). Variable grids (`grid.z= 1 2 4 8 16 ...`) and non-numeric coordinates (`label.earth= Vp Vs density`) can also be defined. A suite of applications may require specific axis names or attributes in order to cooperate, but the DDS architecture does not. The architecture can accommodate unforeseen processing domains and dimension counts as easily and naturally as conventional ones.

Lesson 1: Avoid unnecessary constraints and assumptions.

Generic Format: Traces (binary data) are described by a *generic format*. The name, type and offset of *headers* can be specified, much like *fields* within a C structure. These fields within an *I/O record* can be very simple or arbitrarily complex. In fact, *conventional* formats such as SEGY, SU, SEP, and USP are a subset of the generic format that is supported by DDS (Figure 2). This provides an important foundation for portable software, portable data, operational efficiency and leveraging of non-DDS software.

Generic formats are very flexible. Fields within an I/O record may be a scalar or an array. The `Samples` field is a vector and is handled by DDS the same way as any other header. I/O

records may contain samples without any headers, or just the opposite. Specialized formats can also be created. For example, multi-component traces might have three sample vectors that all share one set of headers.

DDS provides recipes for *creating* formats, such as segy, usp and fcube. The recipes can be adjusted at run time for special cases, without compiling or linking. For example, end users could define "`fmt:*:segy.SEGY_TRACE_SPARE= typedef struct { float4ibm muf; float4 foo; float4x bar; } SEGY_TRACE_SPARE;`". This adjustment adds three custom fields (`muf, foo, bar`) to the segy trace format (union for the spare area). Their types are three flavors (IBM, IEEE big endian, IEEE little endian) of 4 byte floats. This feature allows DDS to handle bizarre variations of SEGY data. Arbitrary formats can be created by end users with other recipe mechanisms.

Lesson 2: Generic formats provide important flexibility.

Header Maps: Header values can be changed when I/O records are read and written. For example, the external format on disk can be converted to an internal format when read. Changes may be as simple as converting the endian of one specific field or as complicated as converting from SEGY to USP format.

DDS provides rules for *mapping* the content of headers. The rules can be adjusted at run time, without compiling or linking. For example, end users could define "`map:segy:usp.SrPtXC= 3.2808 * SrcX`". This rule would convert a source point x coordinate from meters to feet, when changing from SEGY to USP format. In general, the map expression (right hand side) can be a C-like scalar expression. Unfortunately, DDS does not support vector arithmetic. If it did, maps could be applied to sample vectors too, because all fields are *created equal*. Implementation of this powerful concept is planned for PSEIS[5].

Lesson 3: Maps provide important processing flexibility.

Processing: Most DDS applications can directly read data in a foreign format, convert it on-the-fly to an internal format and apply an algorithm to the headers and samples. If the subsequent processing step is not DDS-enabled, the internal format can be converted again as needed before it is written. DDS can also use disk or tape media for storage, or data can be sent through I/O streams (pipes and sockets).

Generic formats, field maps and media flexibility can boost processing efficiency. This is particularly important when individual data sets are measured in tera-bytes. Flexibility can be used to eliminate processing steps, save precious disk bandwidth and reduce peak storage (Figure 3). DDS can also randomly access disk files that have a foreign format. Other processing systems can not, if they rely upon format converters within a pipeline.

Lesson 4: Synergy is increased when individual features are integrated.

Proprietary formats: Data in a vendor format could be directly read and written by DDS applications. Amoco had a developer's license for the CogniSeis DISCO package. DDS was enhanced to use the DISCO I/O routines, but present the data in the generic format. No changes were needed to DDS applications for them to access Yet Another Format (YAF).

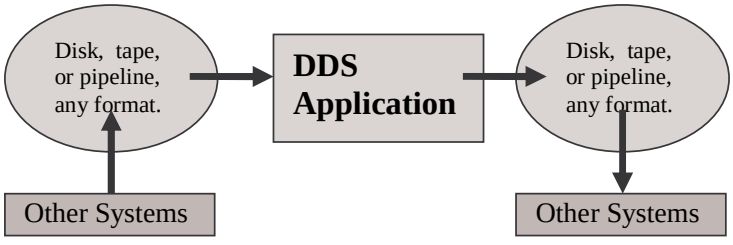


Figure 3: DDS plays nicely with others to increase value.

Care must be taken to avoid litigation when mixing open source and

proprietary software. Mixed licenses raise many questions and are often incompatible. This is one reason that DISCO support was removed from DDS before it was released publicly.

Lesson 5: Know the software pedigree and honor its license.

Emulation: DDS could emulate the API used by other processing systems. For example, in 1998 a wrapper library was created that allowed USP applications to be DDS-enabled, by simply re-linking them. Data was automatically converted between the USP format (used by the application) and other foreign formats. This emulation was surprisingly efficient too. Emulated I/O speed ranged between 90% and 101% of that obtained by the native USP I/O. Yes, for special cases, emulation was *faster* than native, when accessing its own USP format!

Lesson 6: Other systems can leverage functionality provided by DDS.

Information can be lost when DDS converts data between formats. This includes meta-data and trace header information. Foreign formats may lose numerical precision or they may not have a well defined place for certain information. The generic format can typically capture all the information provided by a foreign format, but applications may have difficulty using it. For example, how should a DDS application handle a SEGY "trace identification code" between 9 and 32,767 (non-standard "optional use values")?

Lesson 7: Format conversion and emulation is not a substitute for modernization.

Future: DDS has been successful, but it can be significantly improved. The Parallel Seismic Earth Imaging System (PSEIS) is a new package that will be "OSI Certified open source software[6]". Improvements currently envisioned are described in a companion paper and two poster sessions. "PSEIS, A Processing Architecture Blueprint", describes a collection of enhancements that are designed to increase functionality, convenience and robustness. "PSEIS, Blueprint for Parallel Processing", describes a parallel processing scheme for distributed-memory machines. "PSEIS, Meta-Data Dictionaries" describes a syntax for plain text that has functionality similar to a file system.

Conclusion: DDS provides an efficient, flexible and portable seismic processing architecture that can inter-operate with other processing systems. It has proved its value for research and production work over the last ten years. It is now available as open source software.

Acknowledgment: I'm grateful to the DDS user community within BP and many colleagues from the Amoco Research days for their support. Without the guidance and encouragement of John Etgen, Joe Dellinger, Dan Whitmore and Kurt Marfurt, this project would not have been possible. Special thanks are due Jerry Ehlers for doing a marvelous job of maintaining and enhancing DDS since 1998. Thanks are also due BP America for releasing the DDS software and for permission to publish this paper.

References:

[1] DDS http://FreeUSP.org/DDS/index.html
[2] SEPlib http://sepwww.stanford.edu
[3] CWP/SU http://www.cwp.mines.edu/cwpcodes
[4] FreeUSP http://FreeUSP.org
[5] PSEIS http://PSEIS.org
[6] OSI http://www.opensource.org

W-2 PSEIS, A Processing Architecture Blueprint

*RANDALL L. SELZLER, JOSEPH DELLINGER
RSelzler@Data-Warp.com dellinja@BP.com

Abstract: The Parallel Seismic Earth Imaging System (PSEIS[1]) is a software package that is envisioned as a successor to DDS[2]. It will retain the high efficiency that made DDS suitable for production processing at a leading HPC facility. Its flexibility will continue to facilitate advanced research in seismic processing and imaging. It will also continue to inter-operate with other processing system in order to maximize total value.

The new system provides an alternative to MPI for parallel processing on distributed-memory machines. By tailoring it specifically for seismic processing, applications will be easier to develop, maintain and execute, while remaining efficient and scalable. This scheme is described in a companion poster session "PSEIS, Blueprint for Parallel Processing."

The system supports parallel I/O by slicing data set dimensions across independent file systems. I/O can split headers and samples into separate file for special processing. Two user interfaces are provided that complement each other. Power users can leverage existing skills with scripts and casual users can benefit from the GUI. Object-Oriented technology is used to improve robustness, functionality and long term cost.

Introduction: Data handling is fundamental. Get it right and other issues will fall into place. Get it wrong and the system is doomed to mediocrity.

The PSEIS architecture is similar to that used by DDS. It relies upon the OS to execute independent processes and transfer data among them. Meta-data is kept in plain text dictionaries that can be isolated from the binary data. The binary data is described by multiple dimensions that have extensible attributes. I/O records have a generic format and conventional formats (SEGY, SU and USP) are a supported subset. PSEIS will inter-operate with other processing system using emulation and format conversion on-the-fly. The system will have efficiency, flexibility and portability that is similar to DDS. It will also include basic processing modules that can be used with proprietary ones from third parties.

PSEIS is currently vapor-ware, which has the advantage of being malleable. Colleagues can critique the proposal while the foundation can still be changed. They may also offer important enhancements that are not envisioned here. Participation by others at the design stage may mitigate a not-invented-here syndrome and result in wider acceptance. It may also attract resources for development and maintenance. Vapor-ware also provides a blueprint to guide development by contributors, if the proposal is seriously pursued.

This paper focuses on the extensions in PSEIS that go beyond DDS and why they are important. It also solicits support and collaboration from colleagues and institutions.

Dimension Slices: Binary data is described by one or more logical dimensions. These dimensions can have attributes, such as a name, size and indexing scheme. Additional attributes can be used to describe the *distribution* of I/O records across multiple files. For example, every N_{th} element along a particular dimension can be stored in one of N disk files (modulo distribution, Figure 1). Alternatively, a dimension can be chopped into N blocks and stored in one of N files (tile distribution). Either distribution can be applied to one or more dimensions in the same data set.

DDS supports a simple form of modulo distribution and has proved the value of this concept. For example, migration jobs can drop independent frequency components into a unique data set. These files can be merged into one logical data set for post processing, by simply

tinkering with a dictionary. No binary I/O is required. Similar tricks with the meta-data can be used to logically sort data without any physical I/O. These techniques can be important when dealing with large data sets that tax system resources.

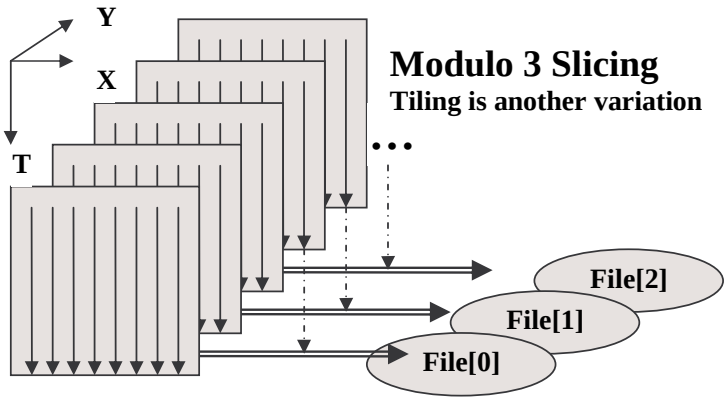


Figure 1: Binary data can be sliced along an axis.

PSEIS can take advantage of the slice concept in additional ways. The individual files can be distributed across independent file systems and the I/O can be performed in parallel. I/O requests can be serviced by daemons that run on the machine that is hosting the file system. The daemons can also mitigate issues associated with network file systems, file locking and deterministic behavior.

Record Splits: I/O records may contain a complex set of headers along with a sample vector. Some processing steps only need a small subset of headers, without any samples. For example, a sort order could be determined by examining only the relevant headers. During this phase, 99% of the data set is baggage that should not be transferred. Later, after the sort order has been determined, the entire I/O record may need to be accessed.

The *split* attribute can be defined for selected fields within an I/O record. Binary data for split fields is kept in a separate file. The application can choose to access the composite I/O record (default) or selected groups of split fields. This mechanism is particularly effective when the processing step that creates a data set also anticipates the subsequent need for split access.

Record splits have several other uses. Fold maps can be derived from header information alone. Some processing systems, such as SEPlib, may require a simple cube of samples. The split mechanisms can park the headers temporarily and attach them again after the cube has been updated.

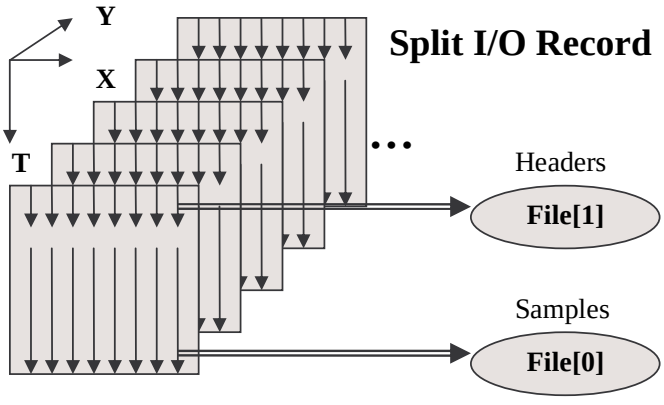


Figure 2: I/O records can be split headers & samples.

Slice and split functionality can be applied to the same data set. Data *bricks* can be created that have N dimensions. Bricks provide efficient access to data that is localized in any combination of dimensions.

Other processing systems may have *specific* applications that contain slice and split functionality. PSEIS includes this functionality in the I/O system. It is available to *all* applications, ensures their *compatibility*, maximizes *code reuse* and can increase *operational efficiency*.

User Interface: DDS only provides a script interface for end users. Such an interface is often preferred by power users. If a GUI were *also* available, then the processing system would be accessible to a wider audience and might attract additional support. If properly designed, the two interfaces could complement each other's functionality and allow both audiences to collaborate.

PSEIS provides both a GUI and script interface (Figure 3). Each job setup is kept in a simple script file. It describes one or more processing steps, data set names and associated parameters. The GUI loads and saves a job setup by scanning and printing the script. Power users can create a script manually or tinker with one created by a GUI. The script must conform to a few simple rules if it will be subsequently loaded by a GUI. Keywords are used to identify critical components. Users can switch between interfaces as needed for a given task.

The script functionality could be provided by several existing languages, including Bash, Csh and Python. We should *not* invent a new one! One, and only one, should be selected to minimize development costs and learning curves. The choice is rather subjective and I prefer Bash. It is simple, functional and stable.

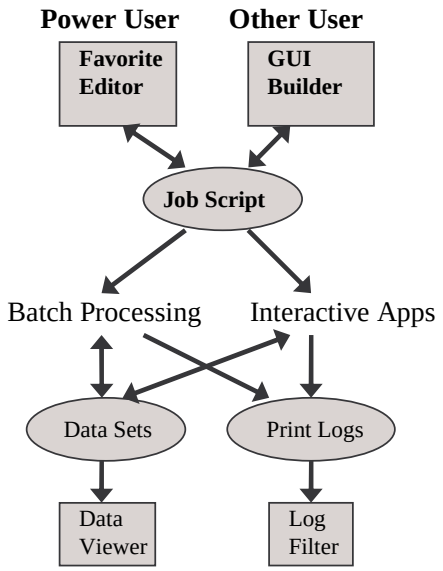


Figure 3: User interfaces and scripts.

The required functionality for a GUI has similarities to graphical builders for software. These tools allow a developer to drag-n-drop widgets (buttons, scroll bars, file selectors) onto a visual canvas. The widgets can be assigned properties (colors, ranges, options) and the interaction between widgets is depicted by connecting lines. Creating a seismic processing flow is another form of high-level programming. The widgets are processing steps (filter, NMO, AGC), properties are parameters (frequency, velocity, increments) and interaction is the data flow (data sets and derived parameters).

PSEIS can leverage the *concept* of GUI programming and *existing tools* that implement it. We should *not* invent something new! Existing open source tools, such as Eclipse and NetBeans, can be extended with seismic module plug-ins. A plug-in is software that bridges the gap between the generalized builder and specific applications. It uses the builder's protocol to list modules, connect widgets, present parameters, change values and preserve states.

Language: The programming language used to implement PSEIS and the API it supports is very contentious. DDS was implemented in C and provided a C and Fortran API. Object-Oriented technology could improve functionality, flexibility, reliability, maintenance and long term cost. Two languages were seriously considered for the architecture itself, Java and Mono. Mono is the open source implementation of Microsoft's C#. Both are appealing because they provide portable, simple and modern support for OOP. Java implementations are more mature and it is currently more popular in the open source community. On the other hand, Mono already provides critical functionality (Table 1) that is still missing from Java, despite serious efforts to enhance it over the last seven years[4].

[4]

Mono (C#)	*Java*	*Critical Functionality,* **Table 1**
Yes	Yes	Jagged arrays (arrays of arrays), float a[i][j][k];
Yes	No	Rectangular arrays, float a[i,j,k];
Yes	No	Array index > 2 giga-elements.
Yes	No	Disable array bound check for efficiency.
Yes	No	Convenient and efficient complex arithmetic.

Mono is my choice, because of the critical functionality it provides and the lack of priority given to Java for scientific computing by Sun Microsystems. The Mono project is sponsored by Novell and is rapidly growing in popularity. It is a very capable OO language that is portable to Linux, Mac OS X, Sun Solaris, BSD and Windows. Projects and people can ride Microsoft's coat tail for installed systems, support packages and training. Novell's Mono home page states "The project implements various technologies developed by Microsoft that have now been submitted to the ECMA for standardization." Microsoft cannot revoke the standards nor the documentation.

The PSEIS API will support Fortran, C and Mono. Application programmers can leverage their existing skills with C and Fortran, while taking advantage of functionality built on OO technology. Programmers that are still *trainable* can further benefit from the Mono API.

Infrastructure: The infrastructure needs to support parallel processing on thousands of nodes. Reporting results to the user in a meaningful way is an essential, but mundane, service. PSEIS uses a *log-filter* scheme, instead of a traditional printout. Individual print lines are grouped into logical entries that are tagged. Tags specify the entry number, time, compute node, processing step, module name, message severity and verbosity level.

The logs can be filtered, based upon tag values, to create a custom report. The original log should include everything that *might* be needed and filters are used to select what is *actually* needed for a particular report. These needs vary with the target audience and their current objective. End users may initially want an error summary, but later want all the details for one particular processing step. System administrators may want usage summaries and programmer may want details on an application fault. All report variations are derived from the same log file. Standard reports could be created automatically for a job and sent to print files or pop-up windows. GUI log viewers could apply filters and change them interactively, in order to drill down to specific details as an investigation unfolds.

The log-filter concept can help programmers resolve difficult problems. Parallel programs may have asynchronous bugs that are difficult to reproduce. Big jobs may be expensive to repeat, just to get more printout. If everything is always captured in the log, post mortems can always be performed if needed. The log messages and their tags provide a rich context that can be used to investigate tricky problems.

Conclusion: Data handling is fundamental. Get it right and other issues will fall into place!

Acknowledgment: I'm grateful to the DDS user community within BP, colleagues from ConocoPhillips and friends from Amoco Research. Special thanks are due Jerry Ehlers, Joe Wade, Rusty Sandschaper and Kyoung-Jin Lee for valuable discussions and support. Thanks are also due BP America for releasing the DDS software and for permission to publish this paper.

References:

[1] PSEIS http://PSEIS.org
[2] DDS http://FreeUSP.org/DDS/index.html
[3] OSI http://www.opensource.org
[4] Java Grande http://www.javagrande.org

W-2 PSEIS, Blueprint for Parallel Processing

*RANDALL L. SELZLER
RSelzler@Data-Warp.com

Abstract: The Parallel Seismic Earth Imaging System (PSEIS[1]) is a software package that is envisioned as a successor to DDS[2]. It provides an alternative to MPI for parallel programming on distributed (and shared) memory machines. The objective is to make it easier to develop, maintain and execute parallel applications that are efficient and scalable. This is accomplished by integrating the technology into the I/O architecture and by tailoring it specifically to seismic processing. The new scheme provides a strategy for fault-tolerant processing, check-point restarts, persistent data storage, out-of-core solvers, data monitoring and parallel debugging.

Introduction: Large seismic data sets often need parallel programs to process them. MPI can work efficiently on distributed-memory machines, but is there something better? Some people, myself included, believe MPI is excessively difficult to learn, program and debug. If a better alternative were available for seismic processing, it could reduce software cost and facilitate the development of parallel programs by a wider audience.

The parallel programming technique proposed for PSEIS is suitable for distributed (and shared) memory machines. It should be efficient, scalable and easy to use for seismic processing. This is a big claim for such a challenging problem!

The basic concept (Figure 1) is to allow PSEIS data sets to use *RAM disk* (real memory) and allow multiple processes to bind offset ranges into their address space. The speed, capacity and functionality of RAM disk is similar to message passing, because all operations are memory based. This scheme leverages the Application Program Interface (API) used for PSEIS data sets. It minimizes the overall learning curve and integrates parallel processing into a seamless architecture. The open-read-write model for data transfer is also familiar to more programmers than the one used by MPI (mpirun, communicators, topologies, tags, messages, envelopes, send and receive).

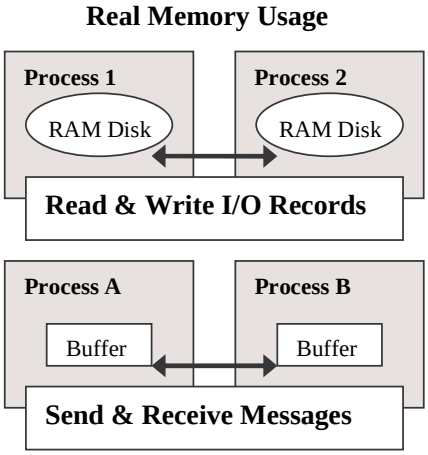


Figure 1: Similarity of MPI and I/O.

The PSEIS mechanism is designed specifically for seismic processing. The generality provided by MPI and byte I/O is desirable when dealing with diverse problems, however it is a liability when dealing with only one specific domain.

Parallel Processing: Parallel processing has three essential elements: *communication*, *synchronization*, and *process control*[3].

Communication: RAM data sets use an extended version of the open-read-write API. After a data set has been opened, a process can *bind* a range of offsets to a buffer in its address space. Other processes can open the same data set concurrently and bind offsets to their own buffers. Subsequent read and write operations within these ranges can be satisfied by a buffer (cache), regardless of where the request was initiated. This functionality is similar to one-sided sends and receives in MPI-2. PSEIS simply leverages a traditional I/O model, rather than imposing MPI's.

The bind mechanism provides important flexibility, convenience and efficiency.

A process can bind a range of offsets that correspond to a specific trace gather or specific elements of an array (like a finite-difference grid). The binding process has *direct access* to the buffer contents and other processes can access it via read and write. I/O requests can be serviced very efficiently, because the memory address for both the source and destination are known when requests are initiated.

N cooperating processes can each bind a *unique* range of offsets and either read (pull) or write (push) data as needed. *Point to point* communication patterns can be established, based upon the unique range associated with each process.

Alternatively, N cooperating processes can each bind the *same* range of offsets. In this case, one process can *broadcast* data to all other processes with one write to the shared range. PSEIS can efficiently and automatically update each cache by using a *propagation tree* to deliver the data.

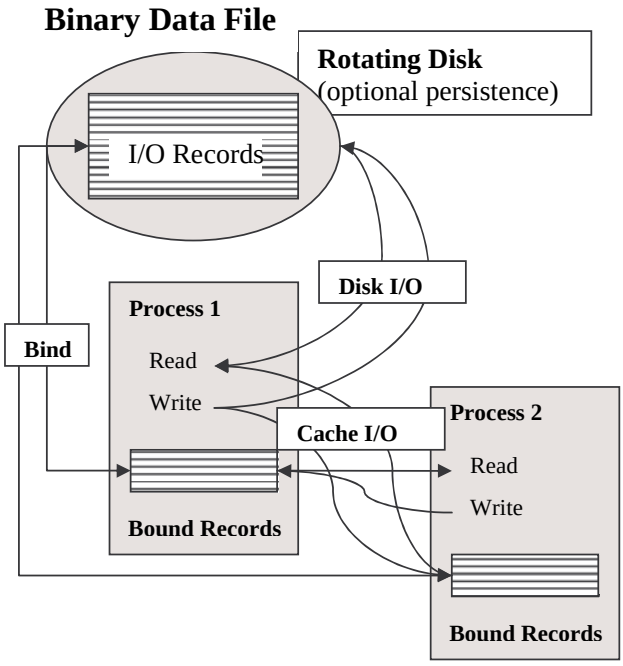


Figure 3: Binding and transferring I/O records.

Binding the same range is also useful when reading data. A *consumer's* read can be satisfied by the first *producer's* cache that becomes available. In this case, the producer's cache must be marked as unavailable (consumed), once it has been paired with a consumer. This mechanism can be used for load balancing and client-server support.

The bind mechanism can be used to implement *restart* functionality. For example, cache buffers can be explicitly *flushed* to persistent storage on rotating disk at *check- points* during processing. This requires one simple call to flush. If sync is also called, processing is protected from system crashes as well. The bind mechanism can automatically use the persistent storage (on rotating disk) to initialize cache buffers upon restart.

Bind can also be used to implement *fault-tolerant* functionality. For example, the broadcast mechanism can be used to preserve state information in independent processes and machines. This redundant state information can be used to automatically start an alternate process, if the primary one fails to respond.

A given process can open a data set and bind zero or more ranges of offsets to buffers. This can be used to *gather* and *scatter* data. For example, N cooperating processes can be responsible for producing (or consuming) every N_{th} trace within a data set. Another process can consume (or produce) a gather by reading (or writing) a contiguous sequence of traces. This can be efficiently implemented, because the scatter-gather pattern is specified once, in advance, and this knowledge can be used for optimization.

The binary format of disk and cache is specified the same way. More importantly, the format is encoded in a meta-data file (dictionary). The file can be read and decoded by other PSEIS

applications. MPI-2 lacks this functionality. The format of messages and I/O records is specified by a sequence of MPI calls. No support is provided for encoding and decoding format information in a meta-data file. Further more, MPI does not allow auxiliary information (units of measure, coordinates and annotation) to be associated with message fields. This makes it difficult for a suite of seismic applications to exchange data, if they use MPI. PSEIS solves this problem by keeping all meta-data about one data set in a dictionary.

The bind mechanism should not be confused with memory mapped I/O on Unix. They both use memory to shadow file contents, but the implementation and functionality is significantly different. Bind is implemented outside the OS using conventional file and socket I/O.

The Unix implementation may impose restrictions on the mapping. For example, the buffer may have to start on a page boundary. Traces and sub-arrays within applications are not generally aligned on pages. Unix maps use virtual memory tricks to identify and process pages that have been accessed or modified. Seismic applications need more explicit control of data transfer, because of efficiency and synchronization requirements. Finally, Unix semantics do not include synchronization of maps across distributed-memory machines.

Synchronization: When a data set is opened (or closed), a network connection is created (or destroyed) to all other processes that currently have the same data set open. These connections are used to transfer data and synchronize activity.

When a data set is opened, a unique *handle* is returned to the caller. Each handle has a *transaction queue*. This queue is used to coordinate activity associated with the data set.

Entries can be pushed onto the queue implicitly or explicitly. For example, a queue entry is *implicitly* created by open and close. Queue entries can also be implicitly pushed by read, write, bind, unbind, flush, sync, reset, lock and unlock. Processes can also *explicitly* push entries (simple messages) to other handles. This can be used to synchronize activity at intermediate levels, signal special conditions, implement barriers and reduce collective operations.

Processes can pop queue entries in FIFO order. Alternatively, they can automatically discard entries based upon their type. Processes must dispose of queue entries in a timely fashion, because it has a finite length. If the queue fills up, the pusher may be suspended, which can delay processing indefinitely.

The list of currently open handles and caches are used to initialize a new queue. Processes can monitor the queue and delay work until all parties have opened the data set (connected). A process can also terminate gracefully, when it sees others close the data (disconnect).

Data transfer and processing can be coordinated using intermediate or low-level synchronization. The required hand shake can be hardwired or negotiated at run time. These details are left to the application programmer and convenience routines that evolve to support them. The scheme is scalable, because communication and control can be decentralized.

Transaction queues can be used to coordinate access of data on rotating disk, not just cache buffers. Software and learning curves for parallel communication can be leveraged to tackle large problems that use out-of-core solutions. A combination of rotating disk and buffer cache can be used to provide a continuous range of solution options.

The transaction queue can also be used to debug parallel programs. Queue entries summarize the interaction between cooperating processes, which is essential knowledge. Entries, serial numbers and time stamps can be monitored in real time or preserved for subsequent analysis. Specialized tools can be created to assist with debugging.

Process Control: Processes can be created on multiple machines using any available mechanism. For example, they could be started manually by a user, launched by a batch

monitor or spawned by a cooperating process. A central authority, like mpirun, is not needed for communication.

Communication is established by opening the same data set. Data sets can be identified by name on a shared file system. Meta-data about an open data set is kept in its *connection* file. If it does not exist, then a new one is created by the first open. If it does exist, network connections are established and a new handle is added to the list of open ones. The open routine implicitly verifies and corrects the connection file. This sequence is reversed when a data set is closed. The last process removes the connection file.

Data sets can also be identified by a URL that references a daemon. The daemon can manage the meta-data and disk I/O for persistent storage. This functionality can be used to implement parallel disk I/O and WAN data access.

PSEIS optimizes communication by accumulating a map of all bound buffers. The system can quickly determine if a particular I/O record is cached, where it is and how to get it. Records may be bound and managed by applications or the *PSEIS I/O system* it self. For example, disk I/O that is cached at one handle can be accessed by another. In this case, the disk I/O system is leveraging parallel communication (which leverages disk I/O, recursively!).

Implementation: Implementing an efficient alternative to MPI is a daunting task, even if limited to a subset of MPI functionality. One approach is to use MPI itself for certain operations. It provides *lots* of flexibility! This could also leverage specialized drivers for high-performance networks. Functionality and licensing issues would have to be carefully investigated to determine whether MPI would be practical.

The PSEIS technology could be implemented using sockets. This might provide a simpler implementation to debug and maintain. It is unclear how much effort would be required to achieve a given level of efficiency.

Topology: An arbitrary topology can exist between cooperating processes and their data sets. The topology can change dynamically while processing. This functionality can be used for load balancing, fault recovery and monitoring purposes. For example, an interactive monitor could open an intermediate data set, even if it only resides in cache. The monitor could capture and display information for QC purposes, without disturbing the nominal processing and flow of data.

Conclusion: Data handling is fundamental. Get it right and other issues will fall into place!

Acknowledgment: Special thanks are due Jerry Ehlers, Joe Wade and Kyoung-Jin Lee for valuable input on this and related projects. Thanks are also due BP America for releasing the DDS software and for permission to publish this paper.

References:

[1] PSEIS http://PSEIS.org
[2] DDS http://FreeUSP.org/DDS/index.html
[3] Timothy Mattson, Beverly Sanders and Berna Massingill. “Patterns for Parallel Programming”. Addison-Wesley, 2004, ISBN: 0321228111.

W-2 PSEIS, Meta-Data Dictionaries

*RANDALL L. SELZLER
RSelzler@Data-Warp.com

Abstract: The Parallel Seismic Earth Imaging System (PSEIS[1]) is a software package that is envisioned as a successor to DDS[2]. Like DDS, meta-data is kept in a plain text dictionary. PSEIS dictionaries also support functionality similar to a Unix file system, but without the overhead. It leverages a familiar and powerful concept for parameter management. A simple, intuitive and familiar syntax is used to encode and decode nested dictionaries within a regular Unix file. This capabilities makes the script interface for power users more convenient and flexible.

Introduction: Dictionaries are used for meta-data, such as processing histories, parameters and binary format descriptions. Dictionaries are part of the interface for end users and programmers. They are encouraged to examine the contents using their favorite text editor. Power users can manually change the contents, if they are careful. This is practical because the text is in a separate file, isolated from the binary data and because it uses a simple syntax. This dictionary concept was borrowed from SEPlib[3] (history files).

Syntax: The dictionary syntax used by DDS only supports a *flat* structure, which makes it difficult to encode and decode some relationships. The syntax used by PSEIS (Figure 1) supports a *hierarchy*. Dictionaries and definitions are analogous to directories and regular files. The new syntax also improves support for comments and definition histories.

The syntax for a definition is a *name*, followed by "=" and then the *value*. The syntax is intuitive, simple and flexible. It is not cluttered with elaborate markup constructs (XML).

3 Way View:

Regular file named "test".

Definition of "test ", value is Vienna.

Dictionary named "test", containing Name, Muf, Bar, Ding, Dong.

Contents of regular file "/home/joe/**test**".

```
Vienna { # Open brace begins "test" Dictionary.
  Name= Value # Optional comment begins with #.
  . . .
  Muf= Austria { # Begin "Muf" dictionary.
    Foo= 1 # Nested definition.
    Bar= 2 # Relative name is "Muf/Bar and
           # full is "/home/joe/test/Muf/Bar".
  } # End Muf dictionary.
  . . .
  Bar= 3.14 # Same name, different name space.
  . . .
  Ding= EAGE 2006 # The one and only Ding.
  . . .
  Dong= oldest value # Dong[3] value.
  Dong= old value    # Dong[2] value.
  Dong= new value    # Dong[1] value.
  Dong= newest value # Dong[0] and Dong value.
  . . .
}  # Close brace ends "test" dictionary.
```

Figure 1: Example dictionary illustrating new syntax.

The value is any sequence of printable characters (except for special ones). It continues across lines until the next definition begins, a nested dictionary is encountered or the enclosing dictionary ends. Special characters within the value must be escaped by prefixing them with "\". These include =, #, $, {, },", [,] and \.

Comments begin with "#" and continue until the end of line. Nothing within them needs to be escaped. They are primarily intended for people, not software. Comments and escape sequences are normally removed from the text before the value is seen by an application.

Applications are free to interpret the value any way they choose. By convention, all white space is equivalent, unless enclosed in double quotes. Quotes are needed for strings that contain white space. New lines within quotes may be ignored by prefixing them with "\".

Nested Dictionary: A definition's value may be followed by one brace pair ("{…}"). If present, it opens the scope for a *nested dictionary*. Each dictionary has a unique name space.

Dictionaries may be nested to any level. The *base dictionary* (which may be a regular file or directory) and its parent are identified by the reserved names "." and "..". The path through Unix directories, dictionary files and nested dictionaries is delimited by "/".

Definition names are relative to a base dictionary, unless they begin with "/". A base is analogous to a current working directory. The base and name are passed to dictionary functions. An absolute base and/or name begins with "/".

```
. . .
axis= x  y  { # Value is list of axis names.
  x= { # Attributes for x axis.
    size= 96
    origin= 0.0
    delta= 50.0
    units= ft
  }
  y= { # Attributes for y axis.
    size= 24
    origin= 1000.0
    Delta= 75.0
    Units= ft
  }
} # End of axis dictionary.
. . .
```

Figure 2: Example value with attributes.

Unlike Unix file systems, each definition can have a value (analogous to a regular file) *and/or* a nested dictionary (analogous to a directory). Nested dictionaries can be used for attributes that further qualify the value. Figure 2 illustrates how they are used to describe data set dimensions and their attributes. Nesting is also a natural way to describe the structure of fields within I/O records.

Names: Definition names may be any sequence of non-white space characters, except =, #, $, {, }, ", [,] and \. By convention, an "=" *immediately* follows the definition name.

History: Processing histories (Figure 3) remember all jobs and steps that have been applied to a data set. The details of each step are isolated in their own nested dictionary. Each is named "`step`". The most recent version can be reference using the name "`step`" or "`step[0]`". Older versions can be referenced using successively larger index values in the C array notation. For example, "`step[0]/args/out`" yields a value of "`bar`", an index of 1 yields "`foo`" and a larger index would yield an empty value.

Functions are provided to count the number of definitions within a dictionary and retrieve their names. Applications can use this information to coordinate work with previous processing steps. For example, the "`equalize`" step could retrieve the "`gain`" used by the "`agc`" step and take appropriate action.

```
step= agc { date=… cwd=… args=… {
    gain= 2
    . . .
  }
}
step= filt { date=… cwd=… args=… {
    out= foo
    . . .
  }
}
step= equalize { date=… cwd=… args=… {
    in= foo
    out= bar
    . . .
  }
}
. . .
```

Figure 3: Illustration of processing histories.

Discussion: Unix file systems do not have a mechanism to retain previous versions of directories and regular files.

Dictionaries do not support the Unix concept of binary data, user IDs, permissions, dates, links and special devices. Dictionaries are kept in plain text within one file. They leverage the concept of a Unix file system, while avoiding overhead that would be required to keep small bits of meta-data in separate regular files.

Dictionary files are convenient for organizing related information. Power users can cut definitions from old histories and paste them into new job scripts. Parameter files can be created or dissected with standard Unix tools such as grep, gawk and sed. This leverages existing software and investments in training.

Conclusion: Data handling is fundamental. Get it right and other issues will fall into place!

Acknowledgment: I'm grateful to the DDS user community within BP and many colleagues from the Amoco Research days for their support. Thanks are also due BP America for releasing the DDS software and for permission to publish this paper.

References:

[1] PSEIS http://PSEIS.org
[2] DDS http://FreeUSP.org/DDS/index.html
[3] SEPlib http://sepwww.stanford.edu

Proposed Open Source Data Processing System

J. Higginbotham, Chevron Energy Technology Company
January, 2006

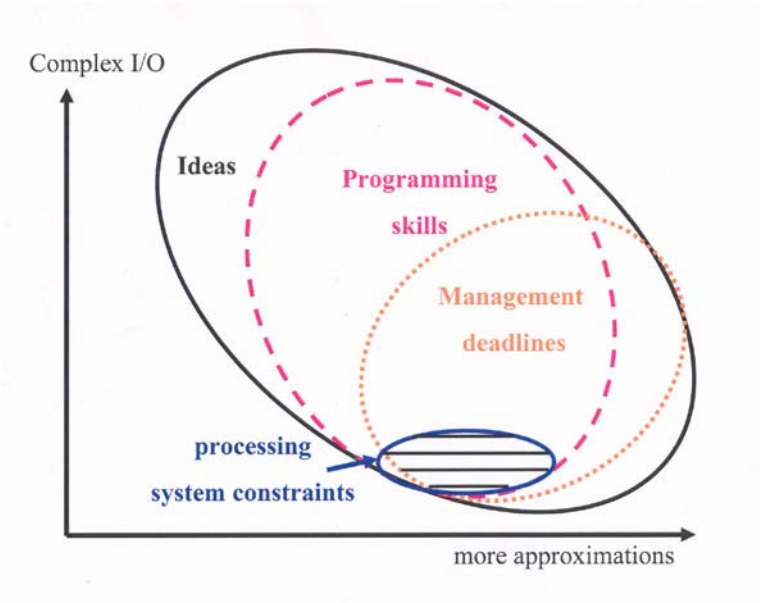


Figure 1: Current constraints on implementation of ideas in production processing systems

Management wants tools developed in the production processing system since they are then immediately available for application. It's hard to argue with this but the fact is that there are constraints placed on research and development by typical production systems. Researchers look at the data processing environment in which they are constrained to work and constrain the ideas they develop to fit that environment ... or create their own environment. This means that many good ideas are discarded up front. The alternative is for the research and development person to write a stand along tool or to spend large amounts of time learning the details of computer science in order to build an environment that minimizes constraints. This is where we are today in the industry. Research geophysicists spend most of their time doing computer science instead of geophysics. If you go from company to company you find that the most complex data processing operations are implemented as either stand alone programs or are part of an originally unofficial, and probably still unofficial, system developed by research people. This is true for both oil companies and contracting companies and is strong evidence that the current situation is as shown in figure 1.

Since the early 1990's there have been great improvements in user interfaces associated with data processing but I've seen little change in the developer interfaces, at least for commercial systems. It would be a mistake to give up these improvements at the user end in favor of changes for the developer. However if the users are to see higher levels of technology available at their friendly end of the processing system then the bottleneck at the development end must be opened. You can easily imagine having systems that fall into three categories: 1. those that serve developers well and users poorly, 2. those that serve users well but developers poorly, and 3. those that serve both users and developers well. When you read my viewpoint below you should remember that while I'd like to put forward something in category 3, as a developer I may unintentionally neglect something important to the user. I suggest though that a weak user interface on an operation in a familiar system is better than a weak user interface on a stand alone program. In any case you should judge what I put forward, take what is useful and combine it with other useful ideas to converge on the best for both users and developers - all ultimately in the interest of users whose work funds us all.

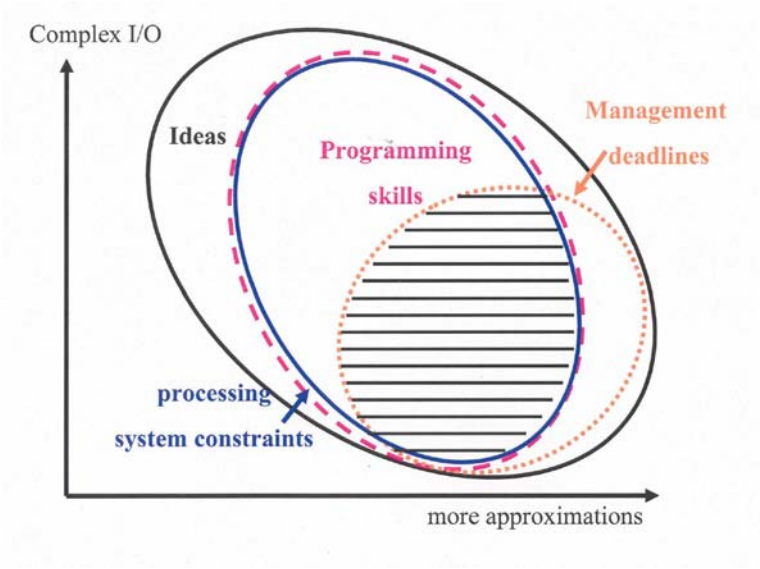


Figure 2: Alleviating constraints on implementation of ideas in an open processing system

The guiding philosophy behind building a processing system that stimulates algorithm development should be the *minimizing of constraints placed on algorithm development.* This should apply to both programs that require interaction with the user throughout the program's activity, interactive processing, and to programs that require only that the user choose a set of parameters that control the action of the program before the program is actually run to process data, batch processing. I am concerned here with batch processing because it is in batch processing that the most complex seismic input-output structures arise.

From the point of view of the developer the seismic processing system's task is to make available data and processing parameter information and to provide a means for passing processed data back to the system. This is a simple task that should not require the developer to become an expert systems programmer. This suggests that the processing system should provide the option of working only with seismic files and parameter argument files that are isolated from the complexities of the system. The benefit of this approach is that the data processing programs running in such a system are not tightly coupled with the processing system. *This will allow the processing system itself to be upgraded or possibly even replaced with minimum or no impact on the data processing programs.* In fact if this interface between the processing system and the data processing applications is sufficiently standardized then *the processing system itself, without the seismic operations, could become a commodity.* Taking this step could require some level of standardization of the seismic data format - I have something in mind of course but there isn't room here.

If I were to stop here I'd have given you some general ideas but not enough specifics to convince you that the ideas were realizable. So what I'll do next is provide you with a rough specification for a system that works. Instead of being formal I'll just draw a diagram and run through what happens when data is processed. It's a prescription kind of presentation but hopefully you'll see how it connects with the ideas mentioned above. And you'll see in the end that a more formal generic specification is available - generic because I'm not pushing my system but rather my ideas.

Consider the system diagram shown in figure 3. The user runs a graphical user interface (GUI) that reads a parameter file ***(P-file)***for every seismic operation that is implemented. The parameter information in the p-file provides the GUI with a list of parameters, how they should be grouped, a one line definition of each, a pointer to more complete documentation, and formatting information that the GUI uses to present the operation and its parameters to the user.

Based on the user's selection of seismic operations to run and their associated parameter information,

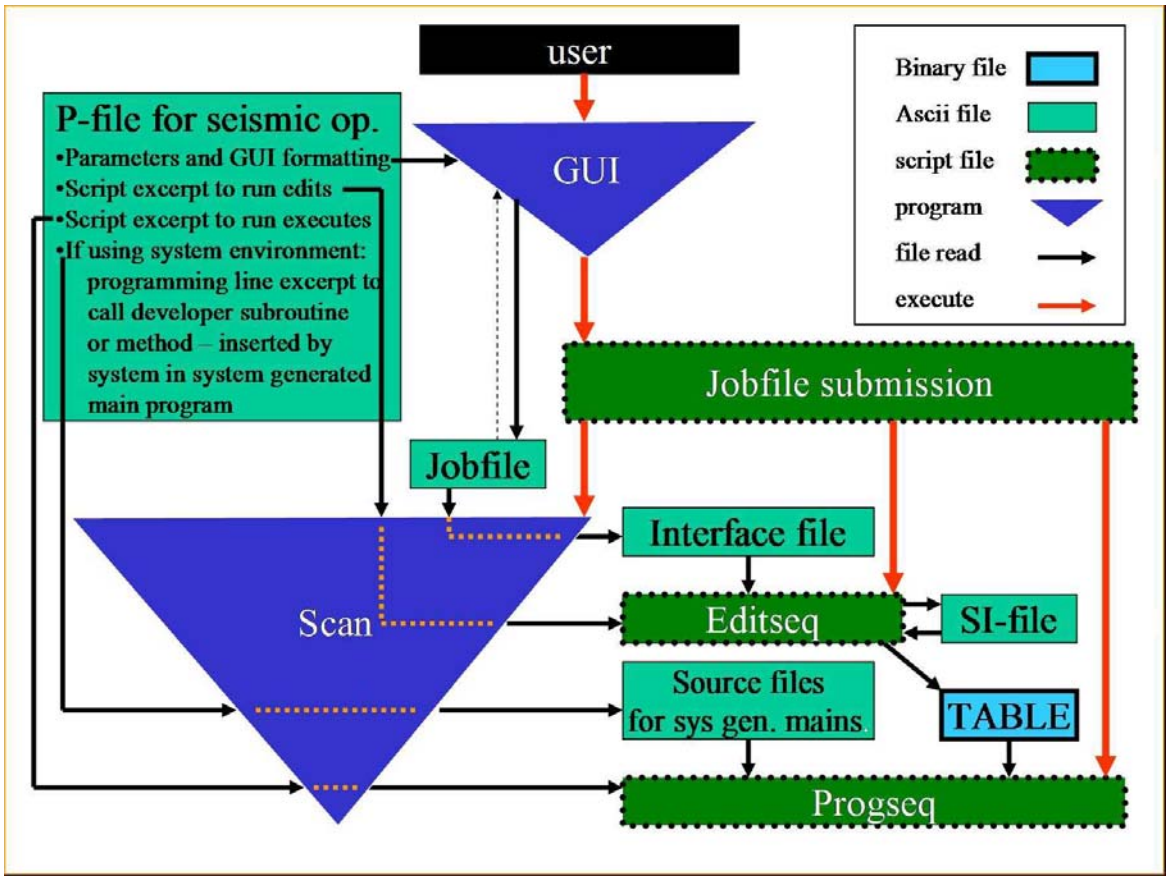


Figure 3: System diagram

the GUI builds an ascii file called a ***Jobfile***. The Jobfile is in a format that can be easily read by the user and can be inserted into reports to document the operations applied to data. The GUI then runs a shell script called the ***Jobfile submission*** script.

The Jobfile submission script runs a program named Scan, a script named Editseq and a script named Progseq. These three run in this order because latter depend on the former.

The ***Scan*** program reads the Jobfile and determines which seismic operations are to execute. As it reads the Jobfile it converts the file to another ascii file, called the ***Interface file***, that is formatted in such a way that its information can be easily read by a computer program without parsing.

By default the seismic operations act on a seismic file named ONLINE. However every seismic operation has access to all seismic files that exist on the file system.

When the scan program determines that a particular operation is to be executed it reads the p-file for that operation and finds the shell script excerpts that are to be placed in the ***Editseq*** shell script. This shell script excerpt will run a stand alone program, called the *edit* for a particular operation. The scan program will have inserted a command into the Editseq script, ahead of the edit program, that will make the portion of the Interface file, corresponding to the particular operation, available to the edit in a file named ***PrmArgs***. The edit will be able to get parameter arguments from the PrmArgs file and check them for errors or process them as required. The processed information will then be written to a binary file named ***TABLE***. The location at which the parameters are to be written is read from the first word of the TABLE

file and this word is updated after the information is written.

When the edit is running it will need information about the seismic files to be processed in order to fully check parameters for accuracy. For those files that exist when the job is submitted this information can be read directly from the file. For files that will be created during the execution of the job this information will not exist. When an edit runs for an operation that will create a new seismic file it must place the fundamental structure information of that file in the Seismic Information file ***(SI-file)***. This is where subsequent edits will look for the information needed for files that don't yet exist.

The Editseq script runs to completion and executes the edit corresponding to each seismic operation to be run before any seismic processing starts. As the edit for a seismic operation becomes mature the probability that that operation will run properly becomes high if the Editseq script completes without errors. One of the last things done by the Editseq script is to reset the pointer in the TABLE file to the location of information for the first seismic operation.

When the scan program identifies a seismic operation to be executed it will also read information from the p-file that controls how the ***Progseq*** script will execute the operation.

The operations most isolated from the system, the ones I'm pushing hard here, are executable programs written by the developer. If this is the type of operation to be run then the scan program will read a script excerpt from the p-file that controls how this executable is to be run and insert it into the Progseq script file. A seismic operation run by Progseq is called an *execute*. The p-file, edit program, and execute program come as a triplet associated with each seismic operation. These are what the developer writes!

The other type of operation that Progseq can run is a seismic operation that is embedded in an *environment created by the system* that serves gathers of traces for processing and takes away gathers after processing. This environment is created inside a stand alone program that is built by the scan program. In a long sequence of operations there may be several of these. Most of this program already exists in a library. Scan only writes a small main program that wraps around calls to the developer routines. Excerpts of code are read by scan from the p-file and inserted into the main program to accomplish the call of the subroutine or method. The scan program will make an effort to group together as many of these types of operations as possible under a single main to minimize read and write to disk. In cases where it is appropriate and where the user requests, the scan program can select a main program that supports automatic parallelization over many slave nodes. Since the system builds the main program, this option for inserting seismic operations within the system places some constraints on the developer! So I'm not selling this one hard - but it is so powerful that it should exist.

Since the seismic operations execute as a sequence of stand alone executables it is possible to insert operating system commands between these operations to link, copy, and move seismic files. Corresponding to each operating system command a program should run at edit time to update the SI-file when seismic data is being moved around, copied or deleted - a job for the scan program.

A number of utility programs should exist: 1) one for reading and making minor modifications to seismic files (the sample interval frequently overflows in SEGY format), 2) one for logically mapping several seismic files to one seismic file, 3) one that will read header information in a seismic file and generate a database, ...

I run a realization of this system almost every day I work. The system has been around since 1984. This year I ran algorithms I wrote in 1984 in the pursuit of development of anisotropic imaging of shot gathers. By publishing the structure of the: Interface file, the SI-file, the TABLE file and the seismic format, this system would become fully accessible to any developer who would be free to develop a seismic operation in any desired language. The system has other desirable features that I don't have room to describe here. I hope that sharing my experience will help promote an open constraint-free system for developers so that users and managers can have what they want.

If you find this system interesting then you may want a more complete generic specification as well as information about the realization I use. I hope to publish these in the future.

Divide Proprietary Dependencies and Perhaps Conquer- The Physical Acoustic Lab Integrated Distributed Lab System

By Justin Hedley
Physical Acoustics Lab at the Colorado School of Mines

The Physical Acoustic Lab (PAL) Integrated Distributed Lab System is an Open Source, perhaps proprietary, platform-*dependent* system for acquiring and viewing data in the PAL lab.

Should an Open Source software system be required to run under an Open Source operating system, such as Linux? I would argue that this is not necessary. Linux may be Open Source, but Open Source software is not necessarily required to always be Linux compatible. Requirements may force *parts* of an Open Source system to run only on proprietary platforms such as Windows using proprietary interfaces such as commercial C libraries; however, using Java and/or Jython, these proprietary parts of the system may be isolated so multiple platforms, including Linux, can share the system.

The Java platform is a programming language and Application Programming Interface (API, i.e., libraries), which runs on a number of operating systems including Linux, Windows, and Mac OS X. Java Native Interface (JNI) allows C/C++ platform-dependent libraries to be accessed directly from Java programs. Now, such platform-dependent parts of the system can be isolated from other platforms such as Linux using Java distributed computing tools such as Remote Method Invocation (RMI).

Jython, a Java implementation of Python, seamlessly integrates a Java API using the simpler language syntax of Python. Extension to a Java system may be implemented in Jython.

The API and development of the PAL lab system will be used to describe how and why such an Open Source system was built. For four types of lab devices, a Polytec™ Vibrometer, a Polytec™ Scanning Controller, a Gage™ CompuScope Oscilloscope card, and a Newport™ Universal Motion Controller, abstractions were developed for the device and its parameters. The abstractions were implemented for four actual lab devices. Communication with the CompuScope was achieved via JNI wrapping of the commercial Windows C interface since no (commercial) Linux drivers are available. Communications with the other devices were achieved via serial port communications using Java Communications (javax.comm) API. Java RMI was used to distribute the devices across the net. Jython was used so others could implement their experiments by extending a Java class. A Java GUI was developed to manage device access and run experiments implemented in Jython. Viewers and data processing tools were developed using the Mines Java Tool Kit, which are incorporated the Jython experiment implementation if desired. An effort will be made to see if this system might help motivate Gage™ to Open Source their Beta Linux drivers, from an apparently abandoned project.

Teaching Tools for Geophysics

By

R. Phillip Bording*, David Churchill, and Sebastian Padina

Department of Earth Sciences
Memorial University of Newfoundland
St. John's, Newfoundland
Canada

Abstract
The use of packaged software is a vital part of the undergraduate/graduate experience in teaching geophysics. But the lack of source code accessibility and further the limitations on changing source code for experiments hinders effective teaching. We are developing a set of open source tools for teaching geophysics. These tools use various programming languages, provide interactive and "batch" computing resources for student projects. Among the programming languages we use; Java, Fortran 90, and C and parallel programming with MPI. We find earth science students lack C++ skills and have limited exposure to programming in general. To overcome these deficiencies computational science students were recruited to provide an influx of knowledge and skills to the geo-type students. Further, the high end visualization hardware has placed a real burden on the available skill base for geophysical research. Hence, we are supporting a full time research associate to develop these large scale visualizations processes. We look forward to the development of open source tools for the educational processes in geophysics as a community wide effort.

I will briefly review a layer based tomography package, 3D visualization tools, model building tools for shot record synthesis, and a basic parallel programming support package for developing parallel codes.

Ray-theory based modelling: software packages of the SW3D consortium

Petr Bulant [1)], Vlastislav Červený [1)], Luděk Klimeš [1)], Ivan Pšenčík [*2)]

1) Charles University, Faculty of Mathematics and Physics, Department of Geophysics, Ke Karlovu 3, 121 16 Praha 2, Czech Republic.

2) Geophysical Institute, Acad.Sci.of Czech Republic, 141 31 Praha 4, Czech Republic. E-mail: ip@ig.cas.cz

SUMMARY

We present description, discussion of advantages and limitations, and examples of applications of FORTRAN ray-tracing program packages, developed within the consortium project "Seismic waves in complex 3-D structures (SW3D)", http://sw3d.mff.cuni.cz. Purpose of the packages is modelling of seismic wavefields in 3-D layered, laterally varying, isotropic or anisotropic structures. The packages are based on the ray theory. Packages are regularly updated. Updated versions are immediately available to the sponsors of the SW3D consortium or to research institutions/individuals who plan to use them for research, and who sign a non-disclosure agreement. Updated versions become public two years after their appearence.

BASIC PACKAGES

The basic ray-tracing packages are CRT (Červený et al., 1988) and ANRAY (Gajewski & Pšenčík, 1990).

Both packages can be used to trace rays and to calculate along them travel times, slowness vectors and dynamic-ray-tracing quantities necessary for calculation of geometrical spreading and vectorial ray amplitudes as well as for two-point ray tracing and other important applications. The calculated quantities can be used to evaluate the elastodynamic ray-theory Green function, and to compute ray synthetic seismograms. Arbitrary types of elementary seismic body waves can be considered: P, S, (in anisotropic layers S1 and S2), direct, multiply reflected, converted. Various types of sources can be situated in arbitrary parts of models. Several acquisition schemes are allowed: surface seismics (land and marine), VSP, cross-hole, OBS, OBC. The packages differ by models of structures, in which considered waves propagate.

Package CRT can be applied, through the use of an independent package MODEL, to very general 3-D layered and block isotropic or weakly anisotropic structures, containing isolated bodies, pinchouts, etc. Inside the layers and blocks, the elastic parameters may vary in all three dimensions. Dissipation and non-planar topography can be considered. The package also allows computations based on the coupling ray theory along isotropic or anisotropic common S-wave rays.

Package ANRAY can be applied to 3-D laterally varying structures containing isotropic and anisotropic non-vanishing layers. The elastic parameters inside layers may vary in all three dimensions. In a smooth model, the package allows S-wave computations based on the quasi-isotropic approximation of the coupling ray theory.

In addition to the above three packages, we also describe several other packages and planned innovations, which include use of Gaussian beams and packets, first-order ray tracing for inhomogeneous weakly anisotropic media, etc.

ACKNOWLEDGEMENTS

The research has been supported by the Grant Agency of the Czech Republic under contracts 205/01/0927, 205/01/D097 and 205/04/1104, 205/05/2182, by the Grant Agency of the Academy of Sciences of the Czech Republic under contract A3012309, by the European Commission under contract MTKI-CT-2004-517242 (project IMAGES), and by the members of the consortium "Seismic Waves in Complex 3-D Structures" (see "http://sw3d.mff.cuni.cz").

REFERENCES

Červený,V., Klimeš,L. & Pšenčík,I., 1988. Complete seismic ray tracing in three- dimensional structures. In: Seismological Algorithms, D.J.Doornbos edit., Academic Press, New York, 89-168.

Gajewski,D. & Pšenčík,I., 1990. Vertical seismic profile synthetics by dynamic ray tracing in laterally varying layered anisotropic structures. *J.geophys.Res.*, **95**, 11301-11315.

Converted wave X-window tools (CXtools) – A practical working example

HENGCHANG DAI* and XIANG-YANG LI
hcd@bgs.ac.uk
British Geological Survey, Murchison House, West Mains Road, Edinburgh EH9 3LA, UK

Summary

A processing package (CXtools) has been developed for multi-component seismic data, based on the new theory of PS converted waves and the standard of Seismic Unix. This package can be used as a standalone package or as a plug-in to Seismic Unix. It extends the ability of Seismic Unix in processing real data. It provides GUI (Graphic User Interface) tools for parameter estimation and parallel prestack Kirchhoff time migration (PKTM) for migration on a PC cluster. It also provides tools for dealing with large datasets and geometry setting. Using this package, we have efficiently processed several marine and land multi-component (3D and 2D) datasets and obtained improved results at low cost.

Introduction

Multi-component seismic data has been widely acquired in recent years and the theory for processing multi-component data has also been developed (Li and Yuan, 2003, Li et al., 2004). However, due to lack of efficient tools to implement the theory, processing the PS-converted wave in multi-component seismic data is more difficult and expensive than processing the P-wave. There is a need for developing appropriate tools based on the new theory for processing multi-component seismic data. Here we present a processing package (Cxtools) we developed for this purpose. This package is intended for processing the PS-converted wave of multi-component seismic data, but it can also be used to process the P-waves. CXtools can be run as a standalone package or as a plug-in to Seismic Unix. In this paper, I will present the principles underlying the development of Cxtools, some examples of Cxtools and the results of processing a real dataset.

Motivation and principles for Cxtools development

The purpose of developing Cxtools is to implement the new theory of PS-converted waves so that we can provide a processing package on Unix/Linux platforms to process multi-component seismic data efficiently and economically. Because developing a complete processing package takes too much effort, we have developed only programs which are not available in other packages. These programs are developed using the same protocols as our chosen packages for I/O and communication, so that they can easily be used with other processing packages. That means the source code of subroutines or standards of the chosen packages for I/O and communication should be openly available. The necessary development tools (compiler and Libraries) should also be freely available. To satisfy these conditions, we chose Seismic Unix as the platform to develop Cxtools.

Seismic Unix (SU) is an open-source package for seismic research, which is developed at the Centre for Wave Phenomena at Colorado School of Mine. It provides the basic programs for researchers to test their ideas. In SU, routines can be connected by the UNIX pipeline facility to perform complicated jobs. Moreover, users can add their own routines into this pipeline. Because SU provides standard input, output and subroutines, researchers can easily develop their own programs according to the SU standard and use them with SU routines. However, since no interactive tools are available for estimating parameters in SU, it is not efficient for processing real datasets and it is difficult to verify the ideas and methods on real datasets. There is a need to extend its ability so that it can be used to process real datasets.

The functions and features of Cxtools

For processing real data, two sets of tools are needed. One set of tools is for processing, for example, tools to perform filtering, NMO, DMO, stacking, migration, etc. Another set of tools is for parameter estimation, for example, tools to estimate the velocity model. The job of estimating parameters is the most labor consuming, and generally needs interactive GUI tools which are not available in SU. Our main task is to develop these GUI tools. The velocity parameters obtained from GUI tools are then passed to the processing tools. Although velocity parameters can be shared in SU routines using the command-line method, it involves much manual editing work and is not efficient. To overcome this, we used a velocity file for sharing the velocity parameters Figure 1 shows the structure of CXtools.

The velocity file consists of the velocity model. For PS converted waves, there are four parameters: the PS converted wave velocity (V_{ps}), the vertical velocity ratio (γ_0), the effective velocity ratio (γ_{eff}), and the anisotropy parameter (χ). For P-wave, we can use the same structure but set γ_0=1.0 and γ_{eff}=1.0. The estimation tools can modify this file but the processing tools can only read it.

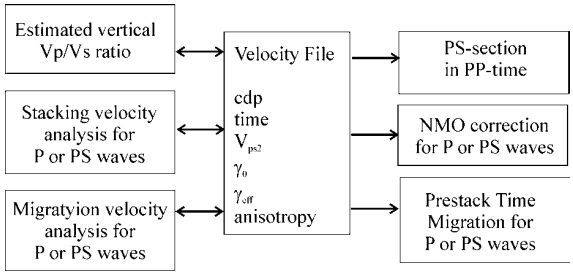


Fig 1. The relationship among the programs in Cxtools.

The Cxtools has the following functions for processing both 2D and 3D datasets:

(1) Estimate vertical Vp/Vs ratio;
(2) Stacking velocity analysis and ACP+NMO+Stacking processing for P and PS waves;
(3) Migration velocity analysis and prestack time migration;
(4) Parallel prestack time migration on PC cluster;
(5) Other routines for handling data, geometry, and velocity files.

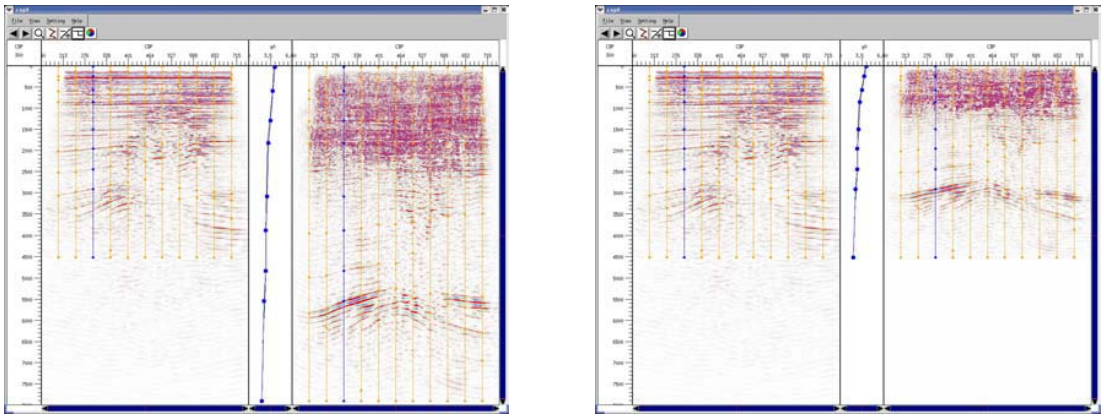
Estimation of velocity is performed using the interactive GUI tools and other functions are performed using non-GUI tools. The parallel version of the prestack time migration is performed on a PC cluster using a Message Passing Interface (MPI). All routines have the same format of input parameters and data I/O as the SU routines, so that the routines of Cxtools can be used with SU routines in the processing flow.

In Cxtools, the most important tools are the GUI tools which are used for estimating velocity parameters. Although one can estimate velocity parameters using SU routines, it is neither efficient nor economical because it involves a great deal of manual editing work. The GUI tools in Cxtools integrate the necessary steps and significantly reduce the processing time and costs. Also the parallel version of prestack time migration is a key routine for processing 3D datasets. It is not practical to perform the 3D prestack time migration for real datasets without the parallel migration programs.

GUI tools for estimating velocity parameters.

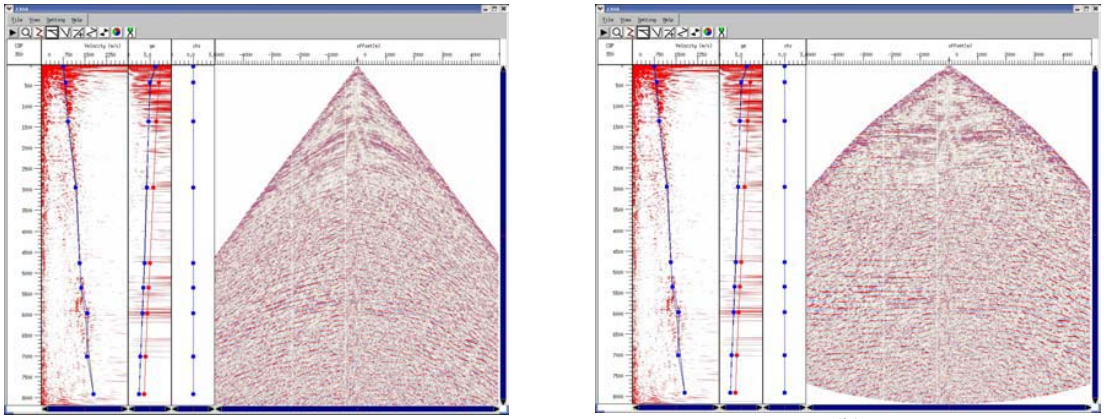
(1) *Estimation of vertical Vp/Vs ratio*. This tool interactively estimates the vertical Vp/Vs ratio (γ_0) by correlating events in the P and PS wave images. Figure 2 shows snapshots of this tool. Using this tool, when the values of γ_0 in the γ_0 panel are picked, guidelines are displayed in image panels to indicate the corresponding events in the PP and PS images. By comparing the events in both images, we can adjust the values of γ_0 so that the guidelines can match the events. The results are then automatically saved to the velocity file.

(2) *Stacking velocity analysis.* This tool interactively estimates the stacking velocity model by applying an anisotropic moveout analysis (Li and Yuan, 2003) to an ACP gather of PS-waves or to a CDP gather of P-waves. Figure 3 shows snapshots of this tool. The velocity spectrum and γ_{eff} spectrum are calculated using the current values of V_{ps}, γ_0, γ_{eff} and χ. When the mouse cursor is picked in the parameter panels, an NMO curve is displayed in the gather panel. We can adjust the values of the velocity parameters so that the NMO curves are matched to the events in the gather. The values of the velocity parameters are then saved to the velocity file. We can also see the results by applying the NMO correction to the gather as in Figure 3 (b). This tool can dynamically show the effects of parameter variation on the NMO correction results.



(a) (b)

Fig 2. Snapshots of the GUI tool for estimating vertical Vp/Vs ratio. The left panel displays the P-wave image and the right panel displays the PS-wave image, against on PS-time in (a) and PP-time in (b). The middle panel displays the values of γ_0 at a CDP location. The blue line indicates the ACP or CDP locations.



(a) (b)

Fig 3. Snapshots of the GUI tools for estimating the stacking velocity model. The left-most panel shows the velocity spectrum obtained from the ACP gather. The blue line indicates the picked velocity. The second panel shows γ_e and γ_0. The blue line shows both values of γ_e and the red line shows γ_0. The third panel shows the anisotropy. The right-most panel shows the ACP gather without NMO correction in (a) and after NMO correction in (b).

(3*) Migration velocity analysis.* This tool interactively updates the migration velocity model by applying a hyperbolic moveout analysis to an inverse NMO CIP gather obtained from anisotropic PSTM. Figure 4 shows snapshots of this tool. The interface of this tool is similar to the tool for estimating stacking velocity (Figure 3), but the gather is the CIP gather obtained from PSTM (Figure 4a) or the inverse NMO CIP gather (Figure 4b). If the events in the CIP gather are not flattened, we can adjust the velocity value according to the inverse NMO CIP gather and perform the PSTM again to flatten the events. This method is described in detail in Dai and Li (2002 and 2003).

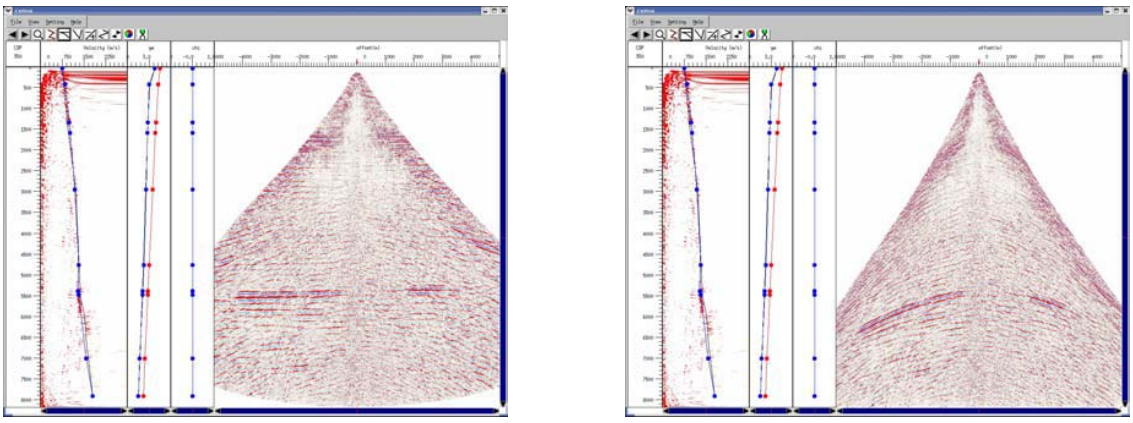
Parallel version of Prestack Time Migration

We also developed a parallel version of the PSTM that runs on a Linux PC cluster. The parallel version has the same I/O as the series version. Therefore we can test the data and parameters of PSTM on a single computer, and once satisfied, apply the parallel PSTM on the PC cluster. This parallel PSTM is vital for processing 3D datasets. The details of the method can be found in Dai (2005).

Applications and Conclusions

Cxtools extends the ability of Seismic Unix to process real data. It can be used as a standalone package or as plug-in to Seismic Unix. The main features of Cxtools are that it provides GUI tools to estimate velocity models, and a parallel tool to perform PKTM on a PC cluster. It also provides tools for dealing with large datasets and geometry setting. The GUI tools significantly reduce the time and

cost of required precisely estimating the velocity model. The parallel version of PSTM greatly reduces the time for migrating 3D datasets. While developing Cxtools, we were able tofocus on implementing the processing theory and interactive functions because we could use the subroutines of Seismic Unix for I/O. EAP and several of EAP's sponsors have installed Cxtools. In EAP we have efficiently applied it to several marine and land (2D and 3D) multi-component datasets and obtained improved results (Dai et al., 2004; Fabio et al., 2005) at low cost.



(a) (b)

Fig 4. Snapshots of the GUI tools for updating the velocity model for prestack time migration. The left-most panel shows the hyperbolic velocity spectrum obtained from an Inverse NMO CIP gather. The blue line indicates the picked velocity. The second panel shows γ_e and γ_0. The blue line shows both values of γ_e and the red line shows γ_0. The third panel shows the anisotropy. The right-most panel shows the CIP gather in (a) and the inverse NMO CIP in (b).

Acknowledgement

This work is funded by the Edinburgh Anisotropy Project (EAP) of the British Geological Survey, and is published with the permission of the Executive Director of the British Geological Survey (NERC) and the EAP sponsors: BG, BGP, BP, Chevron, ConocoPhillips, CNPC, ENI-Agip, ExxonMobil, GX Technology, Kerr-McGee, Landmark, Marathon, Norsk Hydro, PDVSA, Shell, Schlumberger, Total, and Veritas.

References

Dai, H., 2005**,** Parallel Processing of Prestack Kirchhoff Time Migration on a Beowulf Cluster, *Computers and Geosciences*, **31,** No7, 891-899 **.**

Dai, H. and Li, X-Y., 2003, Migration Velocity Analysis of C-Wave using INMO-CIP Gathers of PKTM: A Case Study From the Gulf of Mexico, *Extended Abstracts of EAGE 65th Conference and Exhibition, Stavanger, Norway, 2--5 June 2003, P106*

Dai, H. and Li, X-Y., 2002, A practical approach to update the migration velocity model for PS converted waves, *Extended Abstracts of SEG 72nd Annual Meeting and 2002 International Exposition, Salt Lake City, Utah, 6-11 October 2002, 2227-2230.*

Dai, H., Li, X-Y., and Conway, C., 2004, 3D prestack Kirchhoff time migration of PS-waves and migration velocity model building, *Extended Abstracts of SEG 74th Annual Meeting and 2004 International Exhibition, Denver, CO, 10-15, October,* 2004

Li, X-Y. and Yun, J., 2003, Converted-wave moveout and conversion-point equations in layered VTI media: theory and applications, *Journal of Applied Geophysics,* **54***, 297-318.*

Li, X.-Y., Dai, H. and Mancini, F., 2004, Converted-wave imaging in anisotropic media: An overview, 74th Ann. Internat. Mtg.: Soc. of Expl. Geophys., 881-884.

Mancini, F., Li, X.-Y., Dai, H., and Pointer T., 2005, Imaging Lomond Field using C-wave anisotropic PSTM, *The leading of Edge*, **24**, No 6, 614-616.

Platforms for open-source scientific software

DAVE HALE, COLORADO SCHOOL OF MINES
dhale@mines.edu

Abstract

When developing open-source scientific software, what programming languages should we use? What software libraries should we build upon? What operating systems should we support? And what open-source license should we adopt? Our answers to these and other questions define the platform, and may well determine the success, of our software systems.

When extending an existing software system, these questions have already been answered, perhaps years ago when choices were fewer and simpler. Then, perhaps even our definition of scientific software was simpler. (Did that include parallel computing, or interactive graphics?) But for today's open-source software, we must choose from an increasingly bewildering variety of platforms and consider scientific computing beyond simple array processing.

Today, two platforms are most promising. The first (Java) platform is defined primarily by the Java programming language and the many class libraries that are part of the standard Java runtime environment. The second (C++) platform is defined by a combination of the programming languages C++ and Python, their standard libraries, plus the non-standard Qt or wxWidgets and other libraries.

Both platforms are freely available and widely used for open-source software. Both support multiple operating systems, including Linux, Windows, and Mac OS X. Both support the use of open-source software libraries written in other languages. And both facilitate a broad definition of scientific computing.

However, these two platforms are not equivalent. Furthermore, though both enable the use of software written in other languages, *the Java and C++ platforms are largely incompatible. We must choose one or the other.*

Before choosing, we should compare different aspects of the Java and C++ platforms. Numerical performance is one obvious aspect, but differences here are less significant than we might expect. Other aspects include portability to different operating systems, and the quality and licensing of software libraries. But *simplicity* is the aspect in which these two platforms differ most, and this aspect leads us to favor the Java platform.

Interoperability between open-source seismic packages

Robert G. Clapp and Biondo Biondi, Stanford University, Stanford, CA, USA

Several open source seismic packages now exist and are widely used (SU, SEPlib, FreeUSP, RSF). These packages provide overlapping functionality. In general a user chooses a single package to do most, if not all, of his/her processing. This is not an ideal solution for either the user (he misses out on the functionality that other packages provide) or the package maintainers who must ideally support a huge range of functionality.

The software development process in the open-source community at large provides a good example of the most effective means for a diverse community, with different priorities is to build on each other works. A lesson that they have learned is the importance interoperability. It is useful to be able to seamlessly run programs that come from significantly different code bases.

Most if not all open-source seismic packages treat data as either a series of traces, or as a multi-dimensional hypercube. What we propose is that we as community build and support libraries that provide automatic translation between packages that treat data in a similar fashion. From a user perspective this would allow them to use the best features of each package seamlessly. From a developer perspective they can concentrate on what features they feel are most important, without being concerned about the completeness of their package. We demonstrate this concept by a library that allows SEP3D and SU programs, both using the trace concept, to be run on the same datasets without any code modification.

ANDREWS
ATTORNEYS KURTH LLP

111 Congress Avenue, Suite 1700
Austin, Texas 78701
512.320.9279 Phone
512.542.5202 Fax
andrewskurth.com
ecavazos@akllp.com

Dallas
Houston
London
Los Angeles
New York
The Woodlands
Washington, DC

Open Source E&P Software Licensing Strategies

Edward A. Cavazos

This presentation will explore the legal issues involved in adopting one of various open source licensing models for E&P software and related technologies. The differences between common types of open source licensing schemes (academic, reciprocal, and "standards"-oriented) will be described and a decision tree for selecting the right approach will be set forth, based on the institutional goals of the organization considering the implementation of such a strategy. The presentation will also examine the inherent potential liabilities that accompany open source licensing models and propose mitigation strategies that can be employed to limit the risks. Throughout the presentation, there will be a review of the underlying intellectual property concepts that are pertinent the decisions that must be made in the course of planning and implementing an open source strategy.

Biographies

Stéphane Bellin

Stéphane Bellin is IOR/EOR - Business Development Manager within the E&P Business Unit of IFP. He is in charge of promoting activity dealing with Heavy Oil Recovery, Reservoir Simulation, Reservoir Characterization, Reservoir Monitoring, Well productivity & injectivity. He is also in charge of commercial software development in reservoir engineering.

Biondo Biondi, *biondo@sep.stanford.edu*

Biondo is Associate Professor (Research) of Geophysics at Stanford University Biondo graduated from Politecnico di Milano in 1984 and received an M.S. (1988) and a Ph.D. (1990) in geophysics from Stanford. He is co-director of the Stanford Exploration Project (SEP) and leads its efforts in 3-D seismic imaging. SEP is an academic consortium whose mission is to develop innovative seismic imaging methodologies and to educate the next generation of leaders in exploration seismology. SEP's activities are supported by about 25 companies involved in Oil & Gas exploration and production.

Biondo has made contributions on several aspects of seismic imaging, ranging from velocity estimation to parallel algorithms for seismic migration. Since the early nineties he has been at the forefront of the development of wave-equation 3-D prestack migration methods. In 2004 the Society of Exploration Geophysicists (SEG) has honored Biondo with the Reginald Fessenden Award for his "development of azimuthal moveout (AMO), ... AMO has contributed to many other developments in the areas of migration, multiple attenuation and regularization of seismic data".

Biondo has recently written a book, 3-D Seismic Imaging , that will be the first text book to introduce the theory of seismic imaging from the 3-D perspective. The book will be published by SEG in the Investigations in Geophysics series by the middle of 2006.

Petr Bulant, *bulant@seis.karlov.mff.cuni.cz*

Petr Bulant graduated from Charles University, Prague, Czech Republic in 1994. He received his Ph.D. (1997) from Charles University where he works since 1997. His research interests are in seismic wave propagation, computational seismology and theoretical geophysics, numerical modelling of seismic wavefields and their properties, especially construction of velocity models suitable for ray tracing, two-point ray tracing and travel time calculation, wavefront tracing, weak anisotropy; and in development of seismic software for validation and testing purposes. He works in the research consortium Seismic Waves in Complex 3D Structures. He is a member of EAGE.

Phil Bording, *pbording@mun.ca*

Phil Bording holds the Husky Energy Chair in Oil and Gas Research at Memorial University of Newfoundland, St. John's, Newfoundland, Canada, and long time researcher into seismic modeling and imaging algorithms using the full wave equation.

David Churchill is a research programmer and Sebastian Padina is graduate student in Computational Science at Memorial University.

Edward A. Cavazos

Ed Cavazos is a partner in the Austin office of Andrews Kurth LLP and an adjunct professor at The University of Texas School of Law, where he teaches Software Licensing. His practice involves advising his clients with regard to wide variety of issues including their key strategic transactions, technology licensing, and general technology-related business matters. In particular, Ed regularly handles drafting and negotiating complex licensing agreements (including joint development and OEM deals), technology transfer transactions (including licensing agreements with private and public institutions) and distribution and marketing alliances. He routinely counsels his clients on open source licensing issues, and he represents a number of technology companies developing new open source-based business models. In addition to his transactional practice, Ed has extensive litigation experience in both state and federal courts in matters involving intellectual property, e-commerce and similar issues. Ed is a graduate of the University of Texas (B.A. Philosophy) as well as the University of Texas School of Law (J.D.).

Vlastislav Cerveny, *vcerveny@seis.karlov.mff.cuni.cz*

Vlastislav Cerveny graduated in 1956 and received his PhD in 1962 from the Charles University, Prague, Czechoslovakia. At the same University, he has worked since 1956. His research interests are in seismic wave propagation, particularly in seismic ray method and its extensions. He is a leader of the SW3D Consortium Project.

Glenn Chubak, *gdc178@mail.usask.ca*

Glenn Chubak is a graduate student at the University of Saskatchewan. He obtained a B.A. in Archaeology and a B.S. in Geophysics from the same institution in 2003. His interests include open source software, parallel processing and modelling seismic waves from nuclear explosions. He is a member of the SEG and the CSEG.

Hengchang Dai, *hcd@bgs.ac.uk*

Hengchang Dai is currently a senior geophysicist of the Edinburgh Anisotropy Project in the British Geological Survey. He received his BSc(1982) in Geophysics from The Unuversity of Science and Technology of China, MSc (1984) in Geophysics from Institute of Geophysics, State Seismological Bureau, China, and PhD (1996) in Geophysics from the University of Edinburgh, UK. During 1984-1991, he worked as an assistant research fallow with Institute of Geophysics, State Seismological Bureau. Since 1997, he has been employed by the British Geological Survey. His research interests include seismic anisotropy, multi-component seismology, neural network application, and parallel computing.

Joe Dellinger, *dellinja@bp.com*

Joe Dellinger graduated with a PhD in Geophysics from the Stanford Exploration Project in 1991 and currently works for BP in Houston, specializing in anisotropy and multicomponent seismology. Joe has often provided advice to the SEG (much of it

unsolicited) on how they should best advance into the brave new online / digital world, for which he was awarded Life Membership in 2001. Joe currently is the editor of the Software and Algorithms section of GEOPHYSICS, and maintains the accompanying software and data website software.seg.org.

Sergey Fomel, *sergey.fomel@beg.utexas.edu*

Sergey Fomel is a Research Scientist and Jackson School Fellow at the Bureau of Economic Geology, University of Texas at Austin. He received a Ph.D. in Geophysics from Stanford University in 2001 and was a Postdoctoral Fellow at the Lawrence Berkeley National Laboratory prior to joining the University of Texas. Sergey received J. Clarence Karcher award from SEG in 2001 "for numerous contributions to seismology." His research interests are computational and exploration geophysics, seismic imaging, and geophysical estimation.

Michael E. Glinsky, *michael.e.glinsky@bhpbilliton.com*

Michael E. Glinsky received a B.S. degree in physics from Case Western Reserve University in 1983 and a Ph.D. degree in physics from the University of California, San Diego in 1991. His doctoral research on magnetized pure electron plasmas was recognized by the American Physical Society as the outstanding thesis in the United States (1993 Simon Ramo Award). Before enrolling in graduate school as a National Science Foundation Graduate Fellow, he worked as a geophysicist for Shell Oil Company. After graduate school, he worked as a Department of Energy Distinguished Postdoctoral Research Fellow at Lawrence Livermore National Laboratory for 5 years. More recently he worked for three years at the Shell E&P Technology Co. doing research on Bayesian AVO and 4D inversion. He is currently the Section Leader of Quantitative Interpretation for BHP Billiton Petroleum. He has published over 25 papers in the scientific literature on subjects as varied as plasma physics, signal processing for oil exploration, x-ray diagnostics, application of exterior calculus to theoretical mechanics, and laser biological tissue interactions. He recently received the top Australian scientific medal for his research on petroleum reservoir characterization.

Dominique Guérillot

Dominique Guérillot is Deputy Director of the E&P Centre of IFP since 2004. His areas of professional interest are in Applied Mathematics and Physics, Modelling in Geoscience, Reservoir Engineering and Characterisation, Geology, Up-scaling and Inversion Methods, and Software Development. He is a member of "Society of Petroleum Engineers" and of the "European Association of Geoscientists & Engineers" and in the scientific committee of the Middle East Oil Show and the European Conference on the Mathematics of Oil Recovery. He wrote and contributed to approximately 40 scientific papers and 3 patents.

Dr. James Gunning, *James.Gunning@csiro.au*

James Gunning is senior research scientist in the CSIRO petroleum reservoir characterisation area. He has a BSc(hons) in physics, a BEE(hons) in electrical engineering, and a PhD in Applied Mathematics, all from Melbourne University. His primary interests are in Geostatistics, Geophysics, Inverse problems and applications of

Bayesian statistics to reservoir characterisation problems. These problems always arise in the integration of well, seismic and production data into reservoir models.

Dr Gunning's research experience spans fractal and non-Gaussian
geostatistics, spatial random processes, rock-physics modelling, probabilistic seismic inversion, wavelet and well-tie estimation, and hydrocarbon detection from seismic data. He has expertise in general Bayesian statistics, statistical model building and comparison problems, and modelling of spatially and temporally correlated processes. He is the main author and contact for the open-source Bayesian seismic inversion code Delivery.

Dr. Chris Haase, *Chris.Haase@bhpbilliton.com*

Dr. Chris Haase is with the quantitative interpretation team of BHP Billiton Petroleum in Houston, where he performs quantitative volumetric and risk analyses on current and prospective petroleum assets. More recently, Chris was the lead technology investment principal with the corporate venture arm of BTG International PLC where he was responsible for BTG's high-technology equity investments in North America (the fund has since closed). During his tenure at BTG, Chris led multiple deal teams in the areas of semiconductors, displays, solid state lighting, materials and security and has held board-level positions with portfolio companies. Prior to joining BTG, Chris was a principal at Shell International's private equity division, Shell Technology Ventures in The Netherlands, where he assisted management with strategic business alliances and transactions in the upstream energy sector. Earlier in his career, Chris was a senior scientist at a start-up, Glynn Scientific, and was an adjunct professor at the US Naval Academy where he remains an active member of the Officers' and Faculty Club. In addition, Chris has held several senior positions with various agencies under the US Department of Defense. Chris received his Ph.D. in mathematics from the University of Chicago where he was both a lecturer and a distinguished Department of Defense Fellow and his MBA from Erasmus University, Rotterdam, specializing in strategy and finance. Chris received his Bachelor of Science degree with Honors and Distinction, Summa Cum Laude, from Ohio State University, Phi Beta Kappa.

Dave Hale, *dhale@mines.edu*

Dave Hale received a B.S. in physics from Texas A&M University in 1977 and a Ph.D. in geophysics from Stanford University in 1983. He has worked as a field seismologist and research geophysicist for Western Geophysical, as a senior research geophysicist for Chevron, as an associate professor at the Colorado School of Mines, as a chief geophysicist and software developer for Advance Geophysical, and as a senior research fellow for Landmark Graphics. In 2005, he returned to Mines as the C.H. Green Professor of Exploration Geophysics.

Dave received the Virgil Kauffman Gold Medal from the Society of Exploration Geophysics for his work on dip-moveout processing of seismic data. He also received the SEG's awards for Best Paper in Geophysics in 1992 (imaging salt with seismic turning waves) and Best Paper Presented at the Annual Meeting in 2002 (atomic meshing of seismic images).

Gilbert J. Hansen, *gilbert.hansen@bhpbilliton.com*
Gilbert J. Hansen received a B.S. degree in Physics and M.S. degree in Engineering from Case Western Reserve University and a Ph.D. degree in Computer Science from Carnegie-Mellon University. After graduate school, he taught Computer Science at the University of Houston, Vanderbilt University and the University of Florida. Afterwards, he developed compilers for Texas Instruments and Convex Computer Corp. While at Convex, he managed, designed, and developed three major supercomputer products leveraging advanced research -- vectorizing Fortran and C compilers, an interactive performance analysis tool, and a symbolic debugger for optimized code. After being involved in the R&D of a data-parallel compiler for High Performance Fortran at Rice University, he became a Java Consultant developing applications in the insurance, telecom, data storage, marketing and transportation industries. He recently joined BHP Billiton Petroleum as an IT Global Specialist where he has focused on providing tools to support interpreter's workflow.

Nanne Hemstra, *nanne.hemstra@dgb-group.com*
Nanne Hemstra received an M.Sc. (1999) in Geophysics from Utrecht University in the Netherlands. He joined dGB Earth Sciences in 2001 where he works as a geophysical software engineer. He’s involved in several areas of OpendTect development, like attributes, GUI and visualisation.

Joe Higginbotham, *JHigginbotham@chevron.com*
Joe Higginbotham received a B.A. in Physics from Northern Michigan University in 1971, an M.S. in Physics from The University of Toledo in 1974, and a Ph.D. in Physics from the Uinversity of Toledo in 1977. He worked for Singer Kearfott, an aerospace company, until the fall of 1979 when he took a job with Texaco in Houston. He has worked at Texaco, now Chevron, in seismic processing research, primarily imaging since 1979.

Ludek Klimes, *klimes@seis.karlov.mff.cuni.cz*
Ludek Klimes received an MSc in theoretical physics in 1982, an RNDr in geophysics in 1984, a PhD in geophysics in 1997 and a DrSc in geophysics in 1998, all from the Charles University, Prague. He works within the international research consortium Seismic Waves in Complex 3-D Structures (http://sw3d.mff.cuni.cz) at the Department of Geophysics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic. His research interests involve the theory of seismic wave propagation in heterogeneous 3-D media, with emphasis on various high-frequency asymptotic methods. His main attention is devoted to theoretical development and accuracy estimation of computational methods. He is a member of AMS, OSA, SEG and SIAM.

Igor Morozov, *igor.morozov@usask.ca*
Igor Morozov is a Professor of Geophysics at the University of Saskatchewan (Saskatoon, Canada). He received his M.Sc. in Physics (1982) and Ph.D. in Theoretical Physics (1985) from Lomonosov Moscow State University (Moscow, Russia). Switched to geophysics in late 1980's. Current research interests include analysis of seismic data from Peaceful Nuclear Explosions, nuclear test monitoring, seismic imaging of the crust

and upper mantle, seismic inversion and AVO attributes, integrated and parallel geophysical data processing.

Ivan Psencik, *ivan@ig.cas.cz*
IVAN PSENCIK graduated from Charles University, Prague, Czechoslovakia in 1968. He received his Ph.D. (1974) from the Geophysical Institute of the Czechoslovak Academy of Sciences where he has worked since 1970. His research interests are in high-frequency asymptotic methods, such as the ray method and its modifications, and their applications to seismic wave propagation in laterally heterogeneous isotropic and anisotropic structures. He is a member of the research consortium Seismic Waves in Complex 3D Structures.

Daniel Rahon, *daniel.rahon@ifp.fr*
Daniel Rahon is the Project Manager of the ICarre platform at IFP and software architect for industrial products. As Research Engineer, he published several articles on numerical well testing and identification of geological forms using production data. As IT specialist, he published articles on the introduction of object oriented technologies in scientific software development and recently an article on information technologies for interoperability.

Randy Selzler, *RSelzler@Data-Warp.com*
Randy Selzler is currently President of Data-Warp, Inc. It was founded in 1999 and provides consulting services for seismic imaging software on High-Performance Computers (HPC). Randy has a BSEE from SDSU. He worked for Amoco at the Tulsa Research Center in the Geophysical Research Department for 24 years. This provided valuable exposure to bleeding edge geophysics, seismic processing and the people that make it happen.

Randy's primary interests include advanced processing architectures, High-Performance Computing and seismic imaging applications. He designed and implemented the Data Dictionary System (DDS) in 1995 for Amoco. It was released by BP in 2003 under an open-source license at www.FreeUSP.org. DDS continues to provide the software infrastructure for advanced seismic imaging at one of the world's largest geophysical HPC centers.

John Stockwell, *john@dix.Mines.EDU*
John Stockwell is a graduate of the Colorado School of Mines. He is co-author with Norman Bleistein and Jack K. Cohen of the graduate level textbook _Mathematics of Multidimensional Seismic Imaging, Migration and Inversion_ (2001) Springer Verlag. Mr. Stockwell has been the principal investigator of the CWP/SU: Seismic Un*x project at the Center for Wave Phenomena, Colorado School of Mines since 1996, but has been involved with the project since 1988.