CRS Office: Uma Interface Gráfica para o CRS Convencional

Anderson B. Gomes, Lourenildo W. B. Leite*, Hamilton M. L. Júnior*, Jessé C. Costa*, (*) UFPA*

RESUMO

Este trabalho resume o desenvolvimento do CRS Office que é uma interface Java amigável para o sistema CRS (Superfície de Reflexão Comum), e são apresentados exemplos de aplicações a dados sintéticos e reais da costa Atlântica brasileira. A interface não é limitada aos processos de empilhamento convencional CRS, e a intenção é fazer a integração com outros processos através das ferramentas Java com aplicação centrada no imageamento sísmico com a tecnologia CRS. A interface é extendida para mostrar na tela os resultados obtidos, e para fazer isto é usado o sistema CWP/SU (Center for Wave Phenomena, Colorado School of Mines, Seismic Un*ix) formato e pacotes.

A technologia CRS tem aumentado sua participação no processamento de dados reais nos últimos anos, porém seu uso ainda é restrito a alguns grupos devido, em parte, à não existência de uma interface amigável para tornar seu uso mais eficiente e agradável. Com o objetivo de apresentar uma solução para este assunto, iniciamos o desenvolvimento do CRS Office como uma interface gráfica para o código CRS desenvolvido por Dr. J. Mann e distribuido a partir da Universidade Fridericiana de Karlsruhe, Alemanha. O codigo CRS é escrito em C++, e o CRS Office está sendo desenvolvido como uma interface inteligente no construtor NetBeans (IDE, Integrated Development Environment, 5.5 GUI, Graphical User Interface), na linguagem de programação Java. O CRS Office reúne as vantagens da plataforma Java, como portabilidade e reusabilidade, com a eficiência computacional da programação C++. A interface GUI permite a interação entre o usuário e o código CRS/C++ sem o requerimento de envolver Makefiles e Shell Scripts complexos. No presente estágio, o CRS Office basicamente lê os parâmetros dos blocos (widgets) (TextFields, ComboBoxs, Check Boxs, etc.), realiza o empilhamento CRS baseado em parâmetros de entrada, e imprime os resultados através da criação e execução de Makefiles e Shell Scripts de uma forma transparente. Um próximo e natural desenvolvimento consiste em fazer uma retro-engenharia e uma integração maior entre os códigos do empilhamento CRS e CRS Office usando as ferramentas JNI (Java Native Interface). A situação atual do projeto em pauta é de desenvolvimento, marcado por manutenção, evolução e inclusão de novas ferramentas.

INTRODUÇÃO

Escolhemos dar continuidade na programação Java devido á possibilidade de integrar uma grande quantidade de programas livres de geofísica baseados na plataforma Java, e aos programas livres que já nos são disponíveis. Entre os pacotes de aplicação geofísica baseados na plataforma Java que investigamos, podemos citar Jest (Schwab and Schroeder, 1998) e JavaParty (Philippsen and Zenger, 1998). Jest é um pacote que reúne Jam (Java e matemática) e Jag (Java e geofísica). Jam é uma biblioteca geral e extendível voltada à otimização numérica para ciência e engenharia, e Jag é uma extensão especial de uma estrutura voltada ao processamento e imageamento sísmico. JavaParty é um programa que adiciona transparentemente objetos Java remotos por simples declaração, evitando desvantagens da comunicação direta (socket), evita uma programação extensa do RMI (Remote Method Invocation), e em geral evita muitas desvantagens na transferência de informação. A programação Java-party é especialmente orientada e implementada em sistemas de workstations. Sendo assim, ela combina uma programação tipo Java com o conceito de distribuição de memória dividida em redes heterogêneas. Além disso, Java se apresenta como uma linguagem de aprendizado relativamente fácil e tem uma relação forte com os programas profissionais de processamento geofísico sísmico. Uma interface Java amigável para o sistema CRS, e aplicações a dados sintéticos e reais, é um dos objetivos da nossa participação nos projetos em desenvolvimento, e nas relações intenacionais de cooperação.

CONSTRUINDO CRS OFFICE

CRS Office separa com sucesso as etapas de otimização realizadas pelo código C++ do CRS das aplicaçãoes representadas pelo Java GUI, sem fazer conhecimento de quaisquer Makefiles e Shell Scripts ao usuário. Para apresentar o CRS Office, é conveniente iniciar com algumas informações básicas sobre Java.

Atualmente na ciência e na engenharia novas idéias são implementadas, simuladas e verificadas usando programas de computador. Em geral quando uma nova idéia surge e o desenvolvimento de um novo programa é solicitado, se faz necessário começar a partir de um rascunho, e uma das consequências é que programas anteriores não são úteis para o desenvolvimento de programas futuros. Isto resulta em custo e tempo de desenvolvi-

mento para um novo programa. Pensando nestes dois itens (custo e tempo) foi que idealizamos o CRS Office como uma aplicação Java, uma vez que Java é uma linguagem de programação voltada a objeto (abreviado por OOP, (Savitch, 2006)). OOP é uma metodologia de programação que olha um programa consistindo de objetos que interagem um com os outros por meio de ações. CRS Office foi desenvolvido sob este paradigma, o que faz com que CRS Office seja um código fácil de manter e de reusar.

A plataforma Java é um ambiente de progamação que consiste da máquina virtual Java (VM), e da Interface de Aplicação de Programação (API). A Java API consiste de um conjunto de classes pré-definidas. Qualquer implementação da plataforma Java garante o suporte da linguagem de programação Java, da máquina virtual e da API. CRS Office, como toda aplicação Java escrita na linguagem de programação Java (ver Figura 1), é primeiramente compilada e convertida para um arquivo 'bytecode', uma linguagem intermediária que é a mesma para todos os computadores, e que contêm instruções escritas para a mesma definição de classes ou interfaces em JVM. O arquivo 'bytecode' deve ter um nome idêntico ao nome da classe ou interface definida, e uma extensão do tipo '.class'. Então, o 'bytecode' é compilado na JVM, e a JVM executa suas instruções traduzindo o 'bytecode' na linguagem de máquina para um sistema operacional e 'hardware' particulares. As instuções do CRS Office é para executar o código CRS original através de Makefiles e Shell Scripts. Consequentemente, o código CRS Office é parcialmente portável, até mesmo entre computadores paralelos de diferentes modelos.

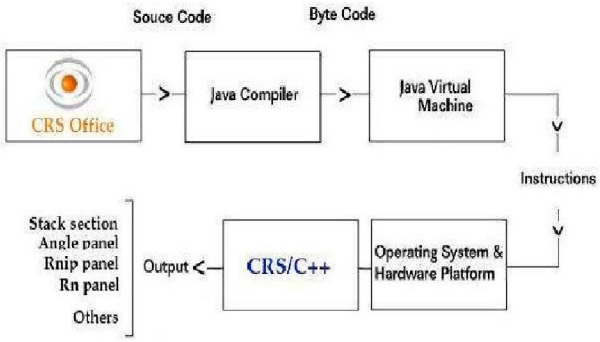

Figura 1: Estrutura de execução do CRS Office. O código fonte do CRS Office é compilado e convertido para um arquivo 'bytecode' que contem a mesma definição de classes ou interfaces escritas em instruções JVM. O arquivo 'bytecode' é compilado na JVM, e a JVM executa suas instruções traduzindo o 'bytecode' para liguagem de máquina para o sistema operacional e plataforma particulares. As instruções do CRS Office executa o código CRS original através de Makefiles e Shell Scripts gerados.

A primeira versão do CRS Office foi desenvolvida no ambiente de aplicações do NetBeans IDE livre disponível na página da Sun Mycrosystem. NetBeans IDE possue muitas ferramentas de produção. Também, o Java permite reunir códigos em programas objetos individuais que promove muitos benefícios, incluindo modularidade, transparência da informação, reuso de códigos, plugabilidade e fácil 'debug'.

CRS OFFICE COMO INTERFACE GRÁFICA DO USUÁRIO

A janela principal do CRS Office é mostrada na Figura 2, e os quadros(widgets) de interação são mostrados na Figura 3, onde o usuário entra com os parâmetros de processamento como mostrado. As informações estão contidas nos quadros da própria figura. A Figura 4 é um exemplo de informação obtida após pressionando o botão Ajuda (Help). Os botões inferiores iniciam com o botão Limpar, seguido pelo botão Executar, depois os botões para retornar e para adiantar quadros, e finalmente com botão para cancelar o processo.

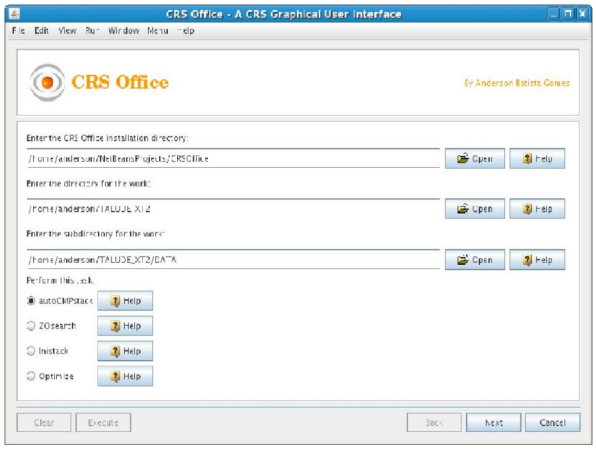

Figura 2: Quadro principal do CRS Office. O painel mostra os campos de textos para entrar no diretório onde o CRS Office está instalado, o diretório onde as linhas para serem processadas estão localizadas, e o sub-diretório onde os resultados serão escritos. As 4 tarefas que podem ser realizadas estão descritas pelos botões Rádio. As janelas de Ajuda (Help) nos botões dão informação sobre a tarefa respectiva.

Figura 3: Segundo quadro de botões do CRS Office. Este quadro mostra apenas se a opção autoCMPstack foi escolhida no quadro principal. Os primeiros 4 campos de textos dão informações auto-explicativa dos conteúdos que são necessários. Os próximos 12 campos de textos (em 2 colunas) são os parâmetros de processamento, onde os botões de ajuda dão as explicações necessárias. O botão Rádio seleciona a condição 'verbose'. A caixa Combo serve para selecionar o tipo de dado para a análise de coerência. Os botões inferiores iniciam com o botão Limpar, seguido do botão Executar, depois os botões para mostrar os quadros anteriores e posteriores, e finalmente o botão para cancelar o processo em execução.

Figura 4: Este quadro é um exemplo da informação obtida após ao pressionar o botão Ajuda(Help), que neste caso é para o parâmetro velocidade da superfície (SURVEL).

Continuando com a interação com os 'widgets' do CRS Office, aparecem as Figuras 5, 6 e 7 se a opção for 'ZOsearch', 'Inistack' ou 'Optimize' no quadro principal. Nestes quadros, os 4 campos-texto do topo dão informações auto-explictivas do conteúdo, e os 16 campos-texto em (2 colunas) são para os parâmeros de processamento, onde os botões de ajuda podem r os conteúdos. Os outros botões inferiores são semelhantes aos do quadro na Figura 3.

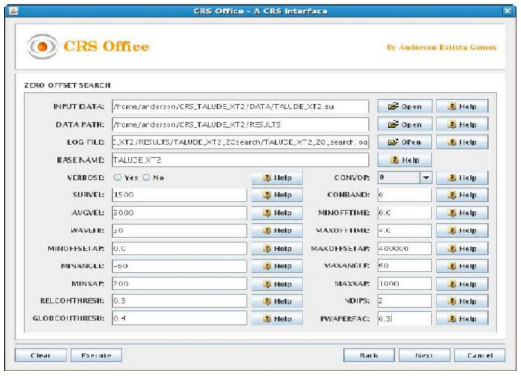

Figura 5: Terceiro quadro de botões do CRS Office. Este quadro aparece apenas se a opção 'ZOsearch' for escolhida no quadro principal. Os primeiros 4 campos-textos superiores dão informações auto-explicativas do conteúdo necessário. Os próximos 16 campos-textos (em 2 colunas) sãos parâmetros de processamento, onde os botões de ajuda dão as explicações necessárias. O botão Rádio seleciona a condição 'verbose'. A caixa Combo serve para selecionar o tipo de dado para a análise de coerência. Os botões inferioes iniciam com o Limpar, seguido do botão Executar, depois os botões para mostrar os quadros anteriors e posteriores, e finalmente o botão para cancelar a execução do processo.

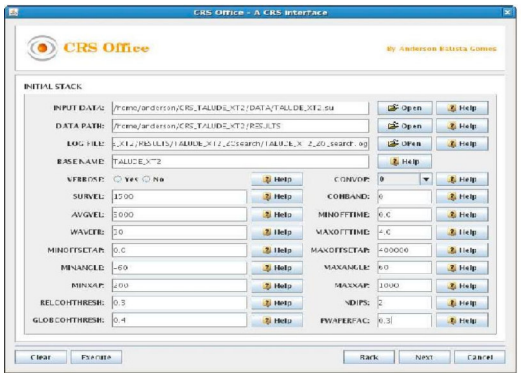

Figura 6: Quarto quadro de botões do CRS Office. Este quadro aparece apenas se a opção 'Inistack' for escolhida no quadro principal. Os primeiros 4 campos-textos superiores dão informações auto-explicativas dos conteúdos necessários. Os próximos 16 campos-textos (em 2 colunas) sãos parâmetros de processamento, onde os botões de Ajuda dão as explicações necessárias. O botão Rádio seleciona a condição 'verbose'. A caixa Combo serve para selecionar o tipo de dado para a análise de coerência. Os botões inferioes iniciam com o Limpar, seguido do botão Executar, depois os botões para mostrar os quadros anteriors e posteriores, e finalmente o botão para cancelar a execução do processo.

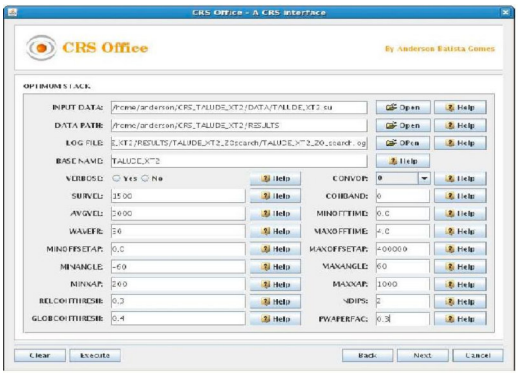

Figura 7: Quinto quadro de botões do CRS Office. Este quadro aparece apenas se a opção 'Optimize' for escolhida no quadro principal. Os primeiros 4 campos-textos superiores dão informações auto-explicativas dos conteúdos necessários. Os próximos 16 campos-textos (em 2 colunas) são os parâmetros de processamento, onde os botões de Ajuda dão as explicações necessárias. O botão Rádio seleciona a condição 'verbose'. A caixa Combo serve para selecionar o tipo de dado para a análise de coerência. Os botões inferioes iniciam com o Limpar, seguido do botão Executar, depois os botões para mostrar os quadros anteriors e posteriores, e finalmente o botão para cancelar a execução do processo.

Pressionando o botão 'Execute', se todos os parâmetros estiverem corretos, um conjunto de procedimentos internos que envolvem a criação de um arquivo *.par*, Makefile e Shell Scripts são realizados para inicializar a execução do código original responsável pelo processamento CRS. Se os parâmetros não estiverem corretos, uma sub-janela independente aparece e uma mensagem de erro informa ao usuário como proceder. A Figura 8 mostra um exemplo de uma mensagem de erro com texto de explicativo.

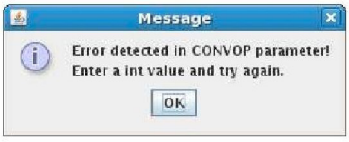

Figura 8: Exemplo de uma mensagem de erro que onde o usuário entrou com um valor inválido para o parâmetro do tipo de dado para análise de coerência (CONVOP).

O CRS Office executa estes Makefiles e Shell Scripts implementando os métodos *getRuntime* e *exec* da classe *Runtime*. A classe *Runtime* se extende através da classe 'Object'. A classe 'Object' é a raiz hierárquica da classe. Cada classe tem a 'Object' como uma superclasse. A classe *Runtime* retorna o objeto associado com a aplicação Java corrente. O método *exec* executa o comando específico num processo em separado. Esta forma conveniente foi a base da classe 'ExecuteCRS' que executa o código CRS nativo. Uma vez que o processo inicia, o usuário pode seguir sua evolução através de um arquivo '.log' como mostrado na Figura 9 com explição no texto da figura.

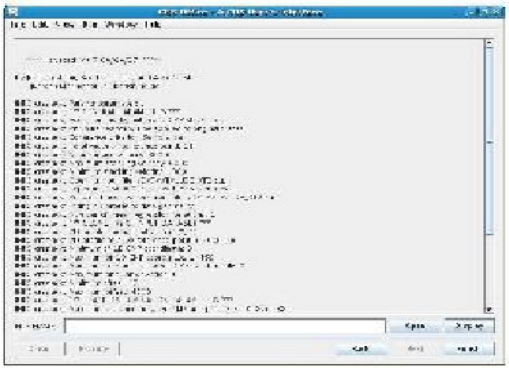

Figura 9: Este quadro mostra o campo '.log' do CRS Office dando informação sobre o processamento. O campo de texto é reservado para o nome do arquivo no diretório 'Results' para uso do utilitário 'suximage' do CWP/SU. O botão 'Open' é para informação, e o botão 'Display' é para olhar a figura.

Uma vez o processo seja concluído com sucesso, o usuário pode escolher um entre os resultados do CRS para mostrar graficamente usando o 'suximage' do CWP/SU. Durante o desenvolvimento da interface, o CRS Office tenderá a ser mais robusto, e dar suporte a mais ferramentas do CWP/SU. A propósito, ele será capaz de dar suporte a todas as ferramentas do CWP/SU; mas, isto não é a intenção atual ao estarmos focados nas ferramentas de visualização do CRS.

SISTEMA CRS

Já consideramos o empilhamento CRS (Mann, 2001) como convencional nas nossas aplicações, e existe uma tendência para demonstrar a complementabilidade do empilhamento CRS aos outros métodos convencionais (como o NMO/DMO) para simular seções afastamento-nulo. Além disso, para cada seção afastamento-nulo são produzidas, em paralelo ao processo de empilhamento, outras seções de atributos cinemáticos do campo de ondas. Como demonstração do processo CRS Office, produzimos resultados para uma linha sísmica marinha sintética e uma linha sísmica marinha real. As figuras (10), (11), (12), (13) e (14) são as seções resultantes do empilhamento, e são, respectivamente, a seção Coerência e as três dos atributos ângulo de emergência, Rnip e Rn para o dado sintético. As Figuras (15), (16), (17), (18) e (19) são para os dados marinhos reais de um conjunto da costa atlântica brasileira.

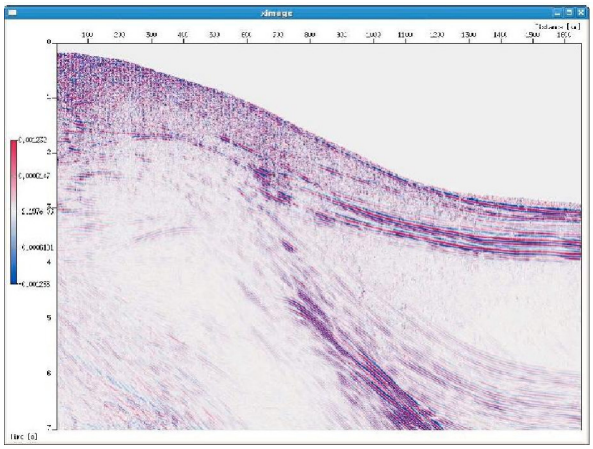

Figura 10: Seção empilhada do dado sintético.

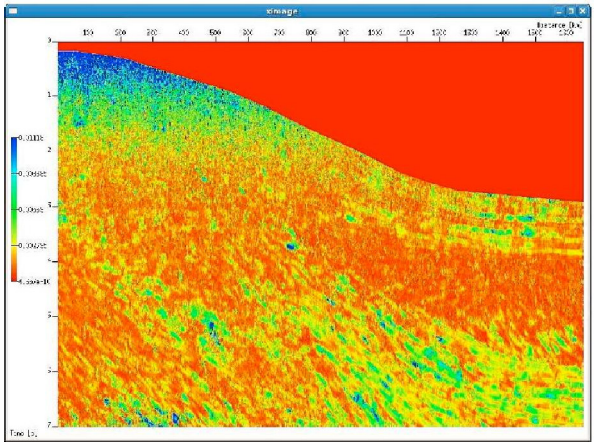

Figura 11: Seção Coerência do dado sintético.

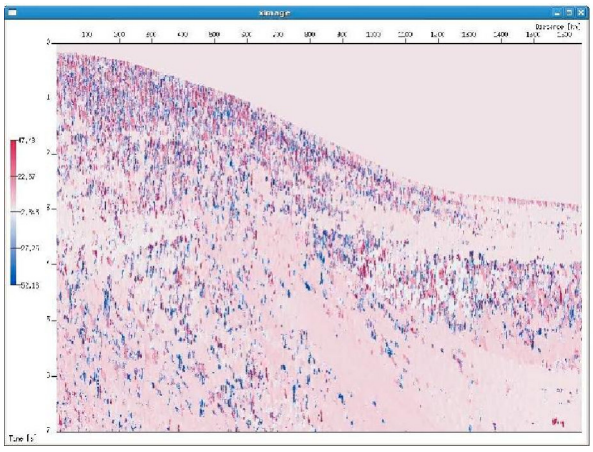

Figura 12: Seção ângulo de emergência do dado sintético.

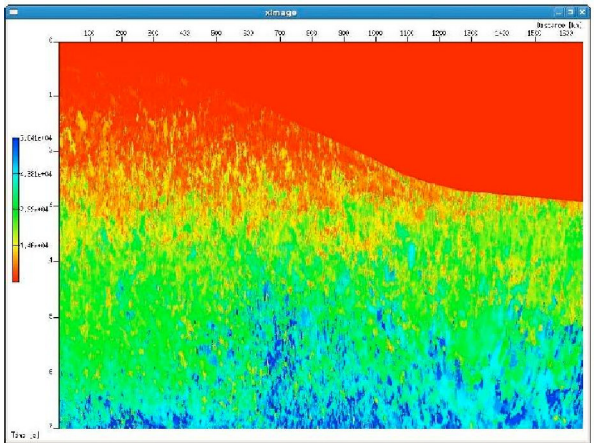

Figura 13: Seção Rnip do dado sintético.

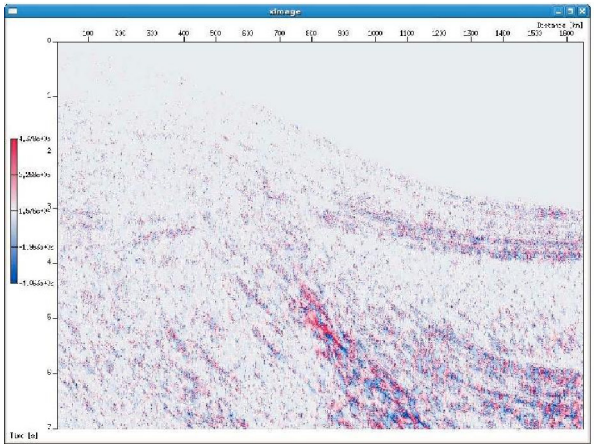

Figura 14: Seção Rn do dado sintético.

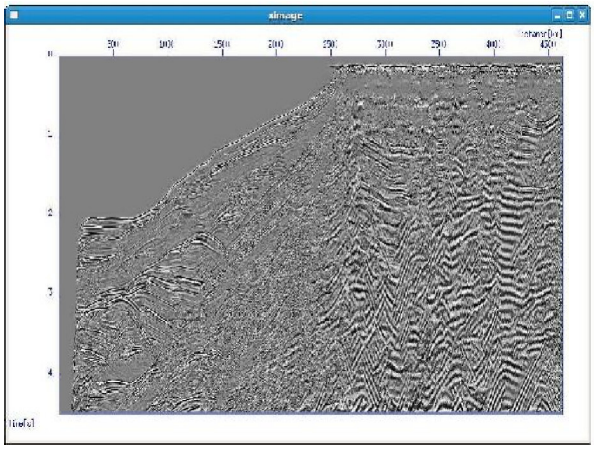

Figura 15: Seção empilhada do dado real.

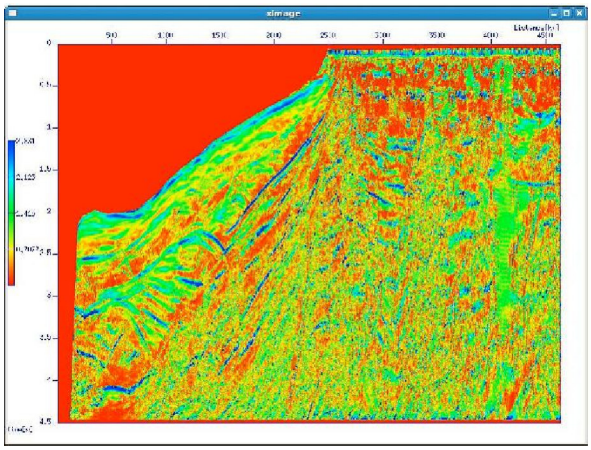

Figura 16: Seção Coerência do dado real.

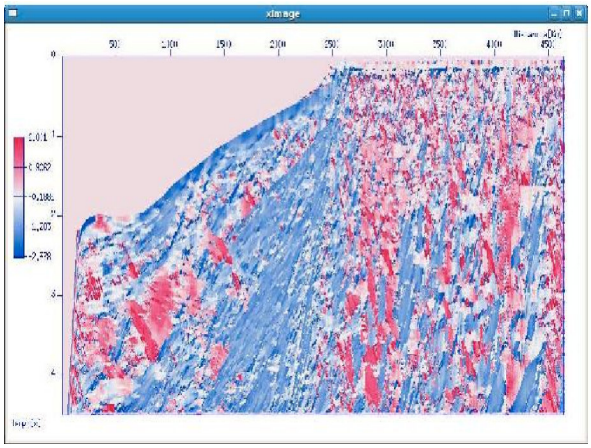

Figura 17: Seção ângulo do dado real.

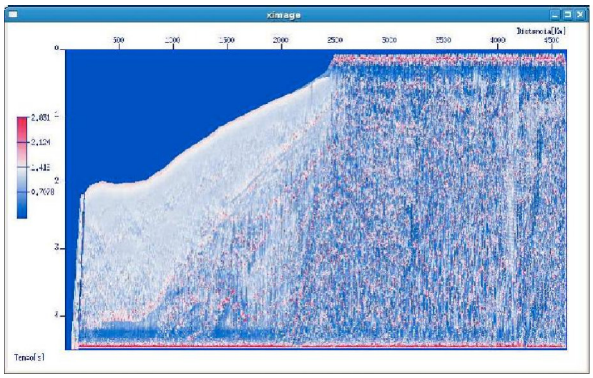

Figura 18: Seção Rnip do dado real.

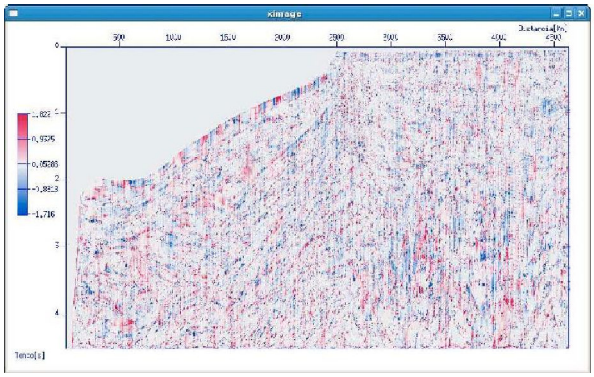

Figura 19: Seção Rn do dado real.

VERSÕES FUTURAS

A primeira versão do CRS Office é ainda relativamente simples, porém já podemos prever com uma forma mais robusta e intuitiva. Entre os desenvolvimentos futuros podemos prever: (1) uso de 'threads'; (2) integração do código CRS/C++ com o CRS Office GUI e JNI; (3) inclusão de mais ferramentas SU; e (4) inclusão de mais ferramentas do sistema CRS.

Um 'thread' pode ser definido de forma simples como um 'stream' de execução que acontece de forma simultânea, independe e em separado a qualquer processo que esteja acontecendo. Um 'thread' é como uma classe de programa que começa num ponto A e entra em execução até alcançar um ponto B, e roda independentemente do que esteja acontecendo no computador. Sem o uso de 'thread' um programa completo pode ser suspenso pelo uso intensivo da CPU, ou por um laço infinito. Com 'threads', as tarefas que não ficam prezas num laço infinito podem continuar o processamento sem esperar pela tarefa em laço terminar. CRS será uma aplicação 'multithread' que permitirá tirar vantagem da linguagem Java. A intenção é permitir que os 'threads' coloquem 'locks' nos recursos compartilhados de forma que, enquanto um um 'thread' está usando o dado, nenhum outro 'thread' pode usar aquele dado, e isto é feito com sincronização.

A Interface Nativa Java (JNI) é um recurso poderoso da plataforma Java. Aplicações que usam a JNI podem incorporar códigos nativos escritos em linguagens de programação tais como C e C++, bem como na linguagem de programação Java. A JNI permite aos programadores vantagens das potencialidades da plataforma Java sem ter que abandonar os investimentos na legalidade do código. A razão é que a JNI faz parte da plataforma Java, e eles podem endereçar operações interativas imediatamente, e esperar que as suas soluções funcionem com todas as implementações da plataforma Java.

DISCUSSÃO E CONCLUSÕES

Cientistas não estão normalmente muito envolvidos em

estabelecer conceitos de 'software' para problemas científicos, uma vez que eles estão mais interessados em usar o 'software' para obter resultados. CRS Office está sendo desenhado para ser uma interface (GUI) consistente, simples e robusta para o empilhamento CRS convencional e seus próximos desenvolvimentos. Complexidade em programas de computador não são bem vindos, e o CRS Office foi desenhado na linguagem Java usando programação orientada a objetos(OOP) para reduzir complexidade pelo uso de classes, métodos e interfaces.

C++ também suporta formalismos orientado-a-objetos, mas entre as linguagens que suportam OOP e satisfazem outros requerimentos de ciência e engenharia, atualmente Java é a mais simples. Além disso, C++ falha em estabelecer um padrão na sua semântica, e se abre principalmente a multi-plataformas e uso genérico. A plataforma Java prover uma enorme biblioteca de classes úteis para uso geral e aplicações individuais. Consequentemente, a escolha do Java foi fundamental para o sucesso do presente projeto.

No presente estágio, o CRS Office se apresenta como uma ferramenta útil e convencional para o empilhamento CRS. O CRS Office está disponível a ajudará e suportará aprendizes preocupados com o processamento do dado, e os fazer descançar com relação aos detalhes computacionais. É sem dúvida importante prover interfaces para fazer com que usuários se concentrem no trabalho de pesquisa, buscando melhores imagens para a subsuperfície baseados na informação do campo de refletividade sísmica.

Este trabalho serve também para demonstrar a colaboração efetiva e as intenções da continuidade de pesquisa em colaboração entre diferentes universidades e, consequentemente, uma forma de se aproximar á indústria para se obter financiamento para trabalhos de pesquisa em tecnologias de exploração de óleo e gás.

AGRADECIMENTOS

Os autores agradecem a FINEP e a PETROBRAS pelo apoio aos trabalhos de pesquisa.

REFERÊNCIAS

Mann, J. (2001). Common-reflection-surface stack: User's manual to version 4.2.

Philippsen, M. and Zenger, M. (1998). Javaparty - transparent remote objects in java. 9(11):1225–1242.

Savitch, W. (2006). Absolute java. 2nd ed.

Schwab, M. and Schroeder, J. (1998). Algebraic Java classes for numerical optimization. 10(11-13):1155–1164.