André Yoshiaki Kashiwabara

Implementação de pipeline de sequenciamento configurável

PIBIC - CNPq

Orientador:
Alan Mitchell Durham

Instituto de Matemática e Estatística
Universidade de São Paulo

São Paulo

2003

Sumário

Lista de Figuras

1 Introdução

Atualmente, uma enorme quantidade de dados relacionados às pesquisas sobre os genomas vem sendo gerada. Para termos uma idéia, a quantidade de caracteres presente no genoma humano pode preencher cerca de 200 listas telefônicas (100 páginas cada) de uma cidade do tamanho de Manhattan[5]. Essas informações devem ser organizadas de forma que possamos encontrar rapidamente os dados relevantes. O desafio presente é o desenvolvimento de projetos de banco de dados para o armazenamento de tais informações e *softwares* para o seu acesso e manipulação.

O laboratório de Biologia Molecular, liderado pelo Prof. Dr. Arthur Gruber da FMVZ-USP, utiliza a Bioinformática para dar suporte aos estudos sobre os aspectos do genoma de *Eimeria* spp. O desenvolvimento de marcadores moleculares do tipo SCAR (*Sequence Characterized Amplified Regions*) gera uma grande quantidade de leituras de DNA que devem ser processadas pelo E-Gene, um sistema modular e configurável para análise de DNA[4]. Após esse processamento, as seqüências são armazenadas no banco de dados *Eimeria* SCARdb, desenvolvida especialmente para armazenar esse tipo de marcador.

Nesse trabalho, implementei o sistema *Eimeria* SCARdb, ajudei a estender o Co-Ed, um editor de configurações gráfico de cadeias de processos. Além de construir o **EGeneDB**, uma base de dados para armazenamento dos resultados obtidos pela execução do *pipeline* de sequenciamento.

2 Objetivos

O principal objetivo deste trabalho foi o desenvolvimento e estudo de ferramentas de bioinformática, de forma a facilitar o processo de análise de dados relativos a pesquisa sobre o genoma.

O trabalho esta dividido em três partes, abaixo segue os objetivos das mesmas.

- **Primeira parte:** Desenvolver a aplicação **Co-Ed** (veja figura 1), um configurador gráfico de cadeias de processos, que serve para configurar um *pipeline* para análise de leituras de DNA.
- **Segunda parte:** Desenvolver um sistema *Web* (*Eimeria* SCARdb), figura 7, para analisar e armazenar os marcadores moleculares do tipo SCAR.
- **Terceira parte:** Desenvolver um banco de dados que possa organizar as informações obtidas através do processamento realizado pelo *pipeline*.

3 Materiais e Métodos

O laboratório contém uma estação de trabalho AlphaServer com o sistema operacional Tru64 UNIX na qual estão instaladas as ferramentas de bioinformática utilizadas ou desenvolvidas pelo nosso grupo.

3.0.1 Desenvolvimento do Co-Ed

O **Co-Ed** é um editor gráfico para configurar um *pipeline* desenvolvido inicialmente pela Flávia Rainone sobre a orientação do Prof. Dr. Alan Mitchell Durham. Ele foi adaptado para facilitar a configuração de parâmetros de componentes de um *pipeline* configurável. Este aplicativo foi desenvolvido sobre a plataforma j2sdk1.4.1 (Java), tornando-o portável.

A seguir vou descrever os principais pacotes do **Co-Ed**, onde podemos observar o uso dos chamados padrões de projeto[2]. Cada padrão descreve um problema que ocorre freqüentemente, assim como a sua solução e conseqüências de seu uso.

3.0.1.1 O pacote DEFSupport

O pacote DEFSupport, implementado pelo aluno Jorge Francisco del Teglia[1], foi usado para implementar um editor de UML e adaptado para fornecer suporte ao *Co-Ed*.

As seguintes classes estão presentes neste pacote.

- **DrawingElement**. Representa qualquer elemento que pode ser desenhado em um painel.

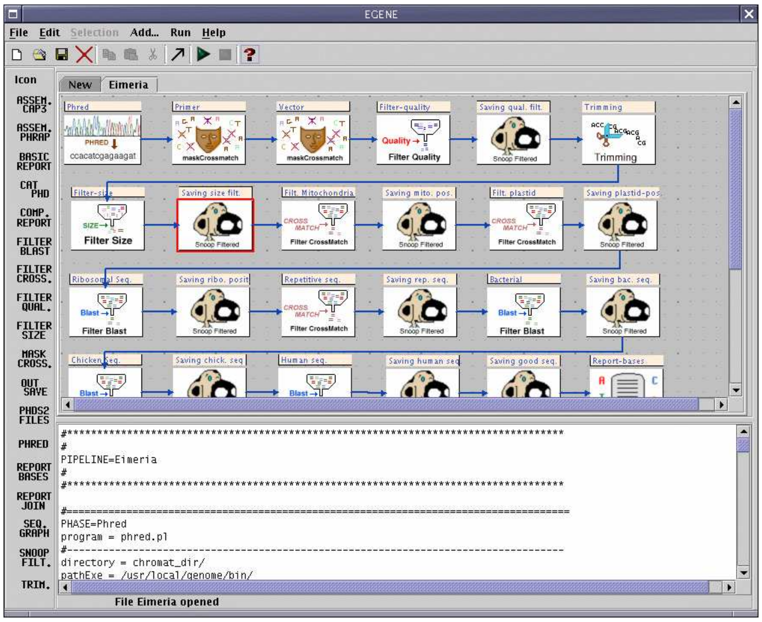

Figura 1: Co-Ed - Configurador do *Pipeline*

- **DrawingElementFinder**. Esta classe é um *Listener* responsável por avisar o cliente sobre ações do *mouse* sobre os objetos da classe *DrawingElement*.

- **AbstractDrawingElementFinderClient**. Representa o cliente do *DrawingElementFinder*, ele é um tipo especial de painel que lida com *DrawingElements*, permitindo que eles sejam adicionados e removidos do painel, assim como selecionados e arrastados

- **DEFBehaviour**. Uma interface que especifica o comportamento de um objeto *DrawingElementFinder*.

- **DefaultDEFBehaviour**. Comportamento padrão do *DrawingElementFinder*, que permite que elementos sejam selecionados e arrastados normalmente, da mesma forma que é feito na maior parte dos programas gráficos.

3.0.1.2 O pacote editor

Neste pacote contém o restante do código, ele é subdivido em outros sub-pacotes. As classes principais dele são:

- **View**. Pode ser considerado a classe mais importante desse pacote, é responsável por acoplar a uma janela principal todas as funcionalidades dos pacotes que são feitos através de *FormattedActions*.
- **ParameterSupplier**. Armazena os parâmetros obrigatórios e opcionais obtidas através de um arquivo de configuração[1].
- **IconData**. Contém uma abreviação do nome do processo que o ícone representa. Podemos gerar os nomes dos arquivos de figuras (.gif) utilizadas para representar esse ícone. Além disso, ela também é utilizada, para obtenção dos parâmetros obrigatórios, opcionais dos programas do sistema de *pipeline*. Esses parâmetros são obtidos através da classe *ParameterSupplier*.
- **ReadEgeneConfiguration**. Um objeto dessa classe lê um arquivo de configuração (.egene) e fornece uma lista de *IconDatas* para o objeto da classe *Model*.
- **Model**. É uma interface que mostra as operações que devem ser implementadas por um modelo desse pacote.
- **MyModel**. É uma implementação do classe *Model*. Ele fornece os objetos *iconDatas*
- **Runner**. É uma classe que executa o sistema de *pipelines* através de um *script Perl*, utilizando o arquivo de configuração gerado pelo editor Co-Ed.

3.0.1.3 O pacote editor.action

O pacote *editor.action* é composto por três classes.

[1] veja arquivo .egene na seção 4.1.1

- **FormattedAction**. Essa classe representa uma ação que pode ser adicionada a qualquer *container* de ações, por exemplo, menu ou uma barra de ferramentas. Esse processo, realizado pela própria classe *FormattedAction*, formata automaticamente o objeto adicionado, criando uma aparência padrão no programa.

- **ToggleAction**. É uma sublcasse de *FormattedAction* e representa uma ação com dois estados. Ambos os estados possui nome, ícones, *tool tips* próprios. Essa ação, quando adicionada a uma barra de ferramentas, apresenta o ícone representante do *firstState*. Quando o botão é pressionado, o estado da ação muda automaticamente, alterando o ícone, o nome, o *tool tip* para o novo estado correspondente.

- **DoubleFormattedAction**. Também é uma sub-classe de *FormattedAction*. Ele representa uma aplicação do padrão *Composite*

3.0.1.4 O pacote editor.graphicView

Esse pacote utiliza o *DEFSupport* para o desenvolvimento de classes que implementam a interface *DrawingElement* e de uma sub-classe de *AbstractDrawingElementFinderClient*.

- **GraphicPane**. Os objetos da classe *DrawingElement* são adicionados no objeto da classe *GraphicPane*.

- **GraphicalElement** é uma implementação do *DrawingElement*.

- **GraphicalTextElement**. é uma sub-classe de *GraphicalElement* e representa várias linhas de texto dentro do retângulo.

- **GraphicalIconELement**. Também é uma sub-classe de *GraphicalElement*. Objetos dessa classe são utilizados para desenhar um ícone na janela.

- **Pipe**. É a representação da flecha entre dois componentes. Nesse caso representa a relação entre dois processos, definindo a ordem dos programas a serem executados no *pipeline*.

3.0.1.5 O pacote editor.graphicView.behaviour

São três as classes que são a implementação de *DEFBehaviour*.

- **CreateIconBehaviour**. Ativado quando o usuário quer adicionar um ícone ao painel.
- **CreatePipeBehaviour**. Ativado quando o usuário quer adicionar um *pipe* ao painel.
- **RunPipeBehaviour**. Ativado quando o usuário quer gerar o arquivo de configuração

3.0.1.6 O pacote editor.memento

Esse pacote aplica o padrão *Memento* através do uso de classes serializáveis do Java.

- **Memento**. Define a interface do padrão *Memento*.
- **GraphicPanePicture**, formada por *Mementos* de elementos, fornece uma maneira eficiente de memorizar elementos e *pipes* de um *GraphicPane*.

3.0.1.7 O pacote editor.memento.document

- **Document**. Implementa o *Memento* e é um *GraphicPanePicture*.
- **DocumentManager**. É responsável pelas ações relacionadas a *Document*, ou seja, abrir, fechar, salvar.

3.0.1.8 O pacote editor.memento.transferArea

Este pacote é responsável por armazenar objetos na área de transferência

- **TransferAreaManager**. É responsável por gerenciar a área de transferência do programa

- **SelectionPicture**. É uma sub-classe do objeto *GraphicPanePicture*, e são manipulados pela classe *TransferAreaManager*.

3.0.1.9 O pacote editor.help

Pacote responsável por exibir a janela de ajuda. Foi utilizado o pacote *javax.swing.help*, fornecido pela Sun.

3.0.2 O sistema *Eimeria* SCARdb

Os SCARs são marcadores de seqüência específica que podem ser utilizados tanto como pontos de referência física do genoma, servindo para propósitos de mapeamento, ou como marcadores genéticos, quando estão associados a algum genótipo de interesse.

O sistema *Eimeria* SCARdb foi criado para organizar os SCARs. No momento, o sistema armazena SCARs específicos para *Eimeria* spp. Porém, o sistema foi desenhado para armazenar SCARs relacionados a qualquer ser vivo, veja figura 2

3.0.2.1 O modelo de aplicação Web utilizado

Uma arquitetura relativamente simples, chamada de *Model 1* [3], foi escolhida. Ele basicamente consiste de páginas JSPs e *Beans*, figura 3. A escolha desse tipo de arquitetura resultou na rápida implementação do sistema. Ele também é de fácil manutenção, já que todos os problemas que ocorrem ficam armazenados em arquivos de *logs*, tornando mais fácil a identificação de erros.

Esse aplicativo Web utiliza o servidor de aplicação Jakarta/Tomcat e banco de dados PostgreSQL.

3.0.3 Desenvolvimento do SequenceObjectDB

O SequenceObject é uma estrutura abstrata de dados que representa uma seqüência. A maior parte dos módulos Perl desenvolvidos no laboratório devem usá-lo quando precisam aplicar alguma operação numa seqüência.

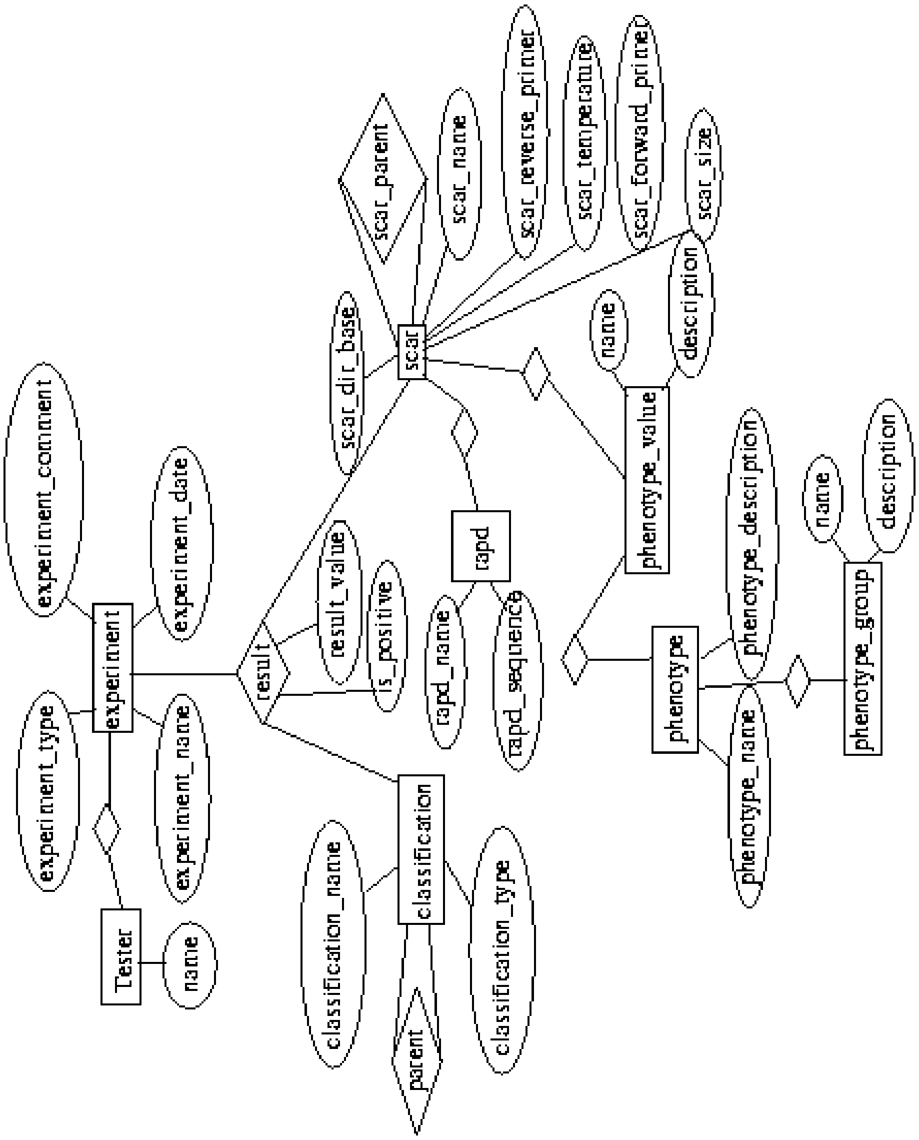

Figura 2: Entidade e relacionamento da base de dados SCARdb

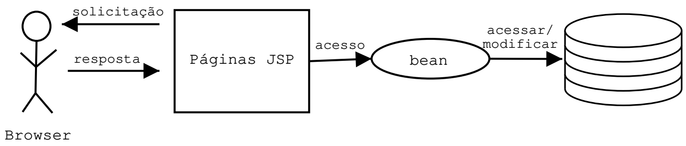

Figura 3: Arquitetura do sistema *Eimeria* SCARdb

As principais operações que o SequenceObject implementa são:

- *read()*, lê uma seqüência de DNA no formato PHD da entrada padrão.
- *print()*, imprime a seqüência de DNA para a saída padrão.
- *trim_left()*, retira uma parte do lado esquerdo da seqüência.
- *trim_right()*, retira uma parte do lado direito da seqüência.
- *mask_current()*, mascara a seqüência corrente.
- *invalidate()*, invalida uma seqüência.

A partir do SequenceObject, surgiu o SequenceObjectDB que implementa a mesma interface do SequenceObject. No entanto, os resultados das operações aplicadas num DNA são armazenados no banco de dados PostgreSQL.

O banco EGeneDB foi modelado segundo a interface do SequenceObject, veja figura 4

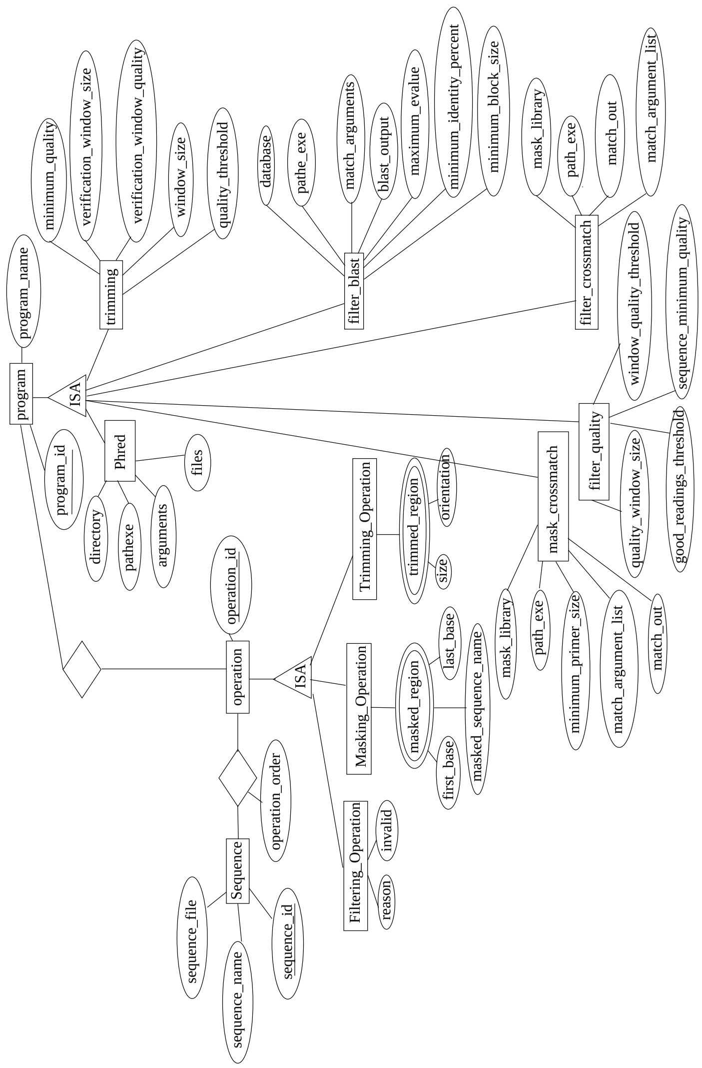

Figura 4: Entidade e relacionamento do banco EGeneDB

4 Resultados e Discussão

4.1 Co-Ed

O uso do **Co-Ed** é intuitivo, veja figura 5. Ele é composto por menu principal, painel de comandos, múltiplas *canvas* para edição, editor de texto e um painel onde estão os componentes configuráveis.

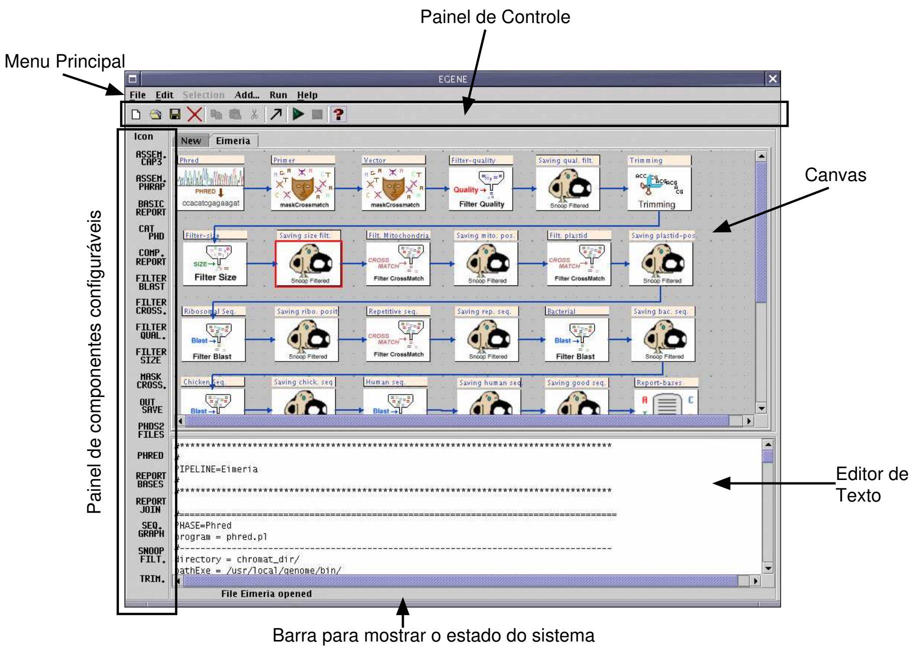

Figura 5: Componentes do E-Gene

Ao adicionar uma nova componente no *canvas*, o Co-Ed pede todos os atributos desse componentes, ou seja, os parâmetros obrigatórios e opcionais da mesma. Por exemplo, ao adicionar o Phred (Representado por Ph) a seguinte janela é apresentada:

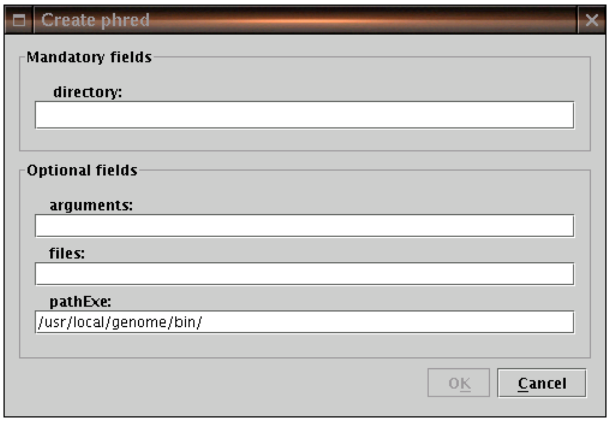

Figura 6: Formulário do Phred

Para cada programa temos um formulário equivalente, o conteúdo do formulário é definido por um arquivo de configuração, o arquivo .egene.

Observe que o botão *OK* não está ativado, isto serve para evitar que o usuário entre com parâmetros inconsistente, este botão é ativado quando todos os campos obrigatórios são preenchidos.

4.1.1 Arquivo .egene

Este arquivo serve para configurar os formulários de cada componente. Na primeira linha, estou indicando que vou configurar o programa phred.pl. Em seguida, linha 2, estou indicando que o "directory" é um parâmetro obrigatório do phred.pl. Nas linhas 3-5, estou falando quais os parâmetros opcionais e os valores padrões dos mesmos. Por fim, indico qual é o prefixo dos arquivos (.gif) que armazenam o ícone do Phred.

```
1 program= phred.pl
2 ==directory
3 pathExe=/usr/local/genome/bin/
4 files=
5 arguments=
```

```
icon = ph

program=trimming.pl
quality_threshold=80
verification_window_size=15
minimum_quality=10
verification_window_quality=20
window_size=30
icon=tr

program=outsave.pl
==savefile
icon=os

program=phds2files.pl
==chromat_dir_base
==save_type
chromat_dir=/dev/null
phd_dir=/dev/null
fasta_file=/dev/null
qual_file=/dev/null
icon=p2

program=filter-quality.pl
percentage_readings_in_sequence=0
minimum_quality_other_bases=10
percentage_good_bases_in_window=95
minimum_quality_in_window=15
window_size=200
icon=qf
```

```
program=catphds.pl
==multiPhdsFile
icon=cp

program= filter-blast.pl
==database
\program=blastn
pathExe=/usr/local/genome/bin/
matchArguments=-W 0
maximum_Evalue=1e-20
minimum_block_size=90
minimum_identity_percent=88
blast_output=/dev/null
icon=fb

program= mask-crossmatch.pl
==maskLibrary
matchArgumentList=-minmatch 35 -penalty -1 -minscore 30
pathExe=/usr/local/genome/bin/
matchOut=/dev/null
icon=mc

program= assemble-cap3.pl
==chromatDirBase
==prefixName
==assemble_base_dir
pathExe=/usr/local/genome/bin/
argument=
report_file=/dev/null
assemble_phds=/dev/null
icon= 3a
```

```

program= filter-crossmatch.pl
==maskLibrary
matchArgumentList=-minmatch 35 -penalty -1 -minscore 30
pathExe=/usr/local/genome/bin/
matchOut=/dev/null
icon=fc

program= snoopFiltered.pl
==program
==output_file
format_file=phd
library=.*
valid=false
icon=eye

program= basicSequencingEffectivenessReport.pl
==report_file
icon=eyeStop
look_stop

# program= juntaRelatorios.pl
# ==file_with_filenames
# icon=jr
```

Observe que podemos ter comentários, veja as três últimas linhas, elas começam com o símbolo #, ou seja, as palavras que estão na mesma linha e que aparecem depois do símbolo # são para serem descartadas durante o parseamento desse arquivo.

4.2 Eimeira SCARdb

4.2.1 Interface *Web*

A interface *Web* consiste de três partes. A primeira é simplesmente uma página de apresentação, a segunda parte serve para consultar o banco e a última serve para atualizar o banco.

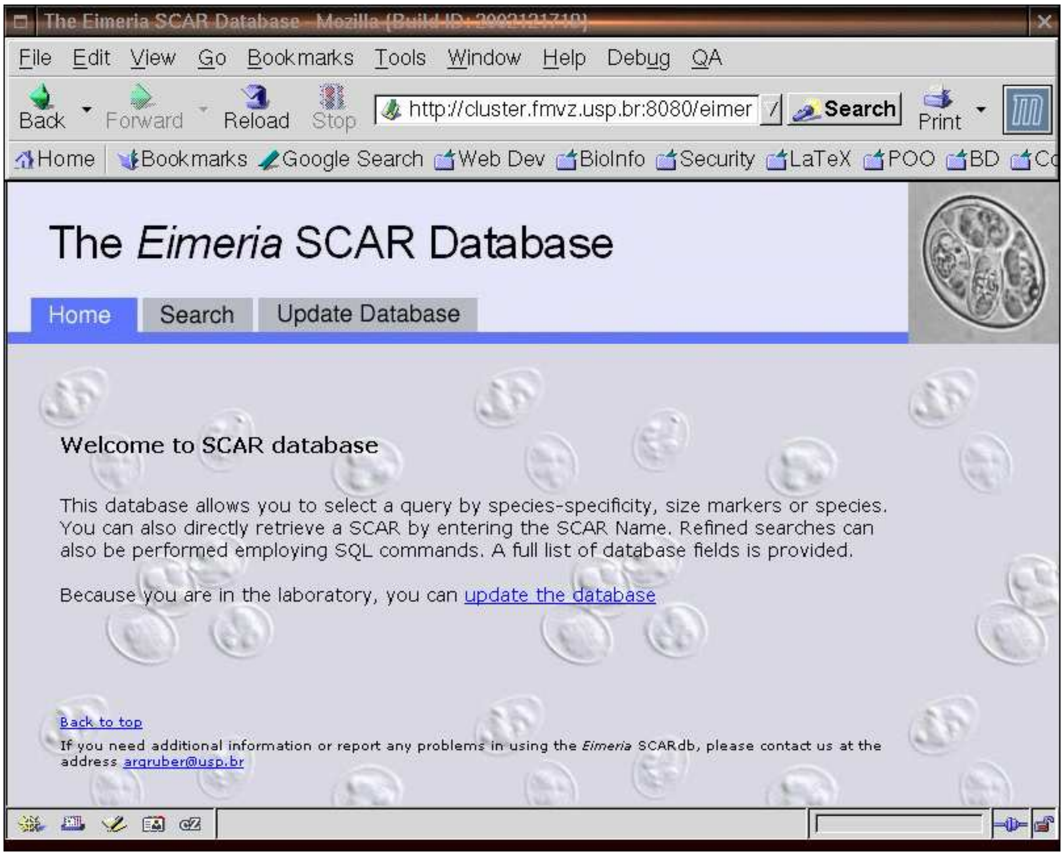

Figura 7: O banco *Eimeria* SCARdb

4.2.1.1 *Search*

De acordo com a figura 8, é possível consultar o banco de três modos diferentes. Na primeira busca, podemos selecionar um conjunto de SCARs a partir da espécie, do tamanho do SCAR, da sua especificidade e do tipo do SCAR (*full* or *nested*). Na seguda busca, simplesmente procuramos o SCAR pelo seu nome. Na última, podemos enviar comandos SQL para obtermos buscas personalizadas.

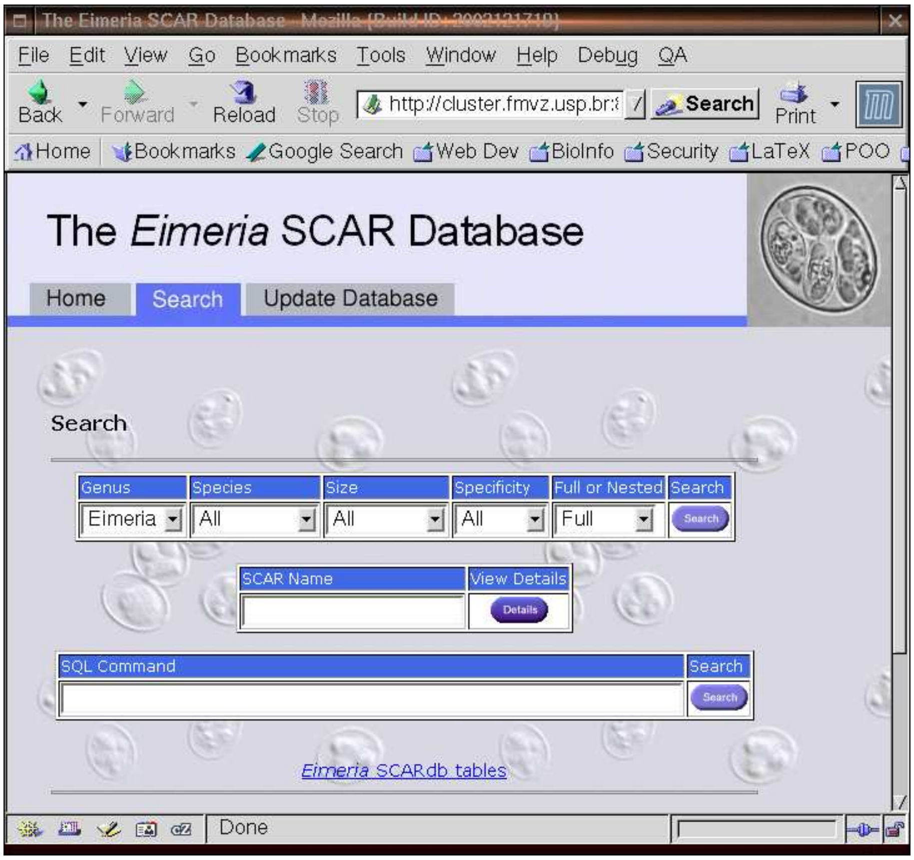

Figura 8: Buscando por SCARs, seção *Search*

Um resultado possível da primeira busca é mostrado na figura 9

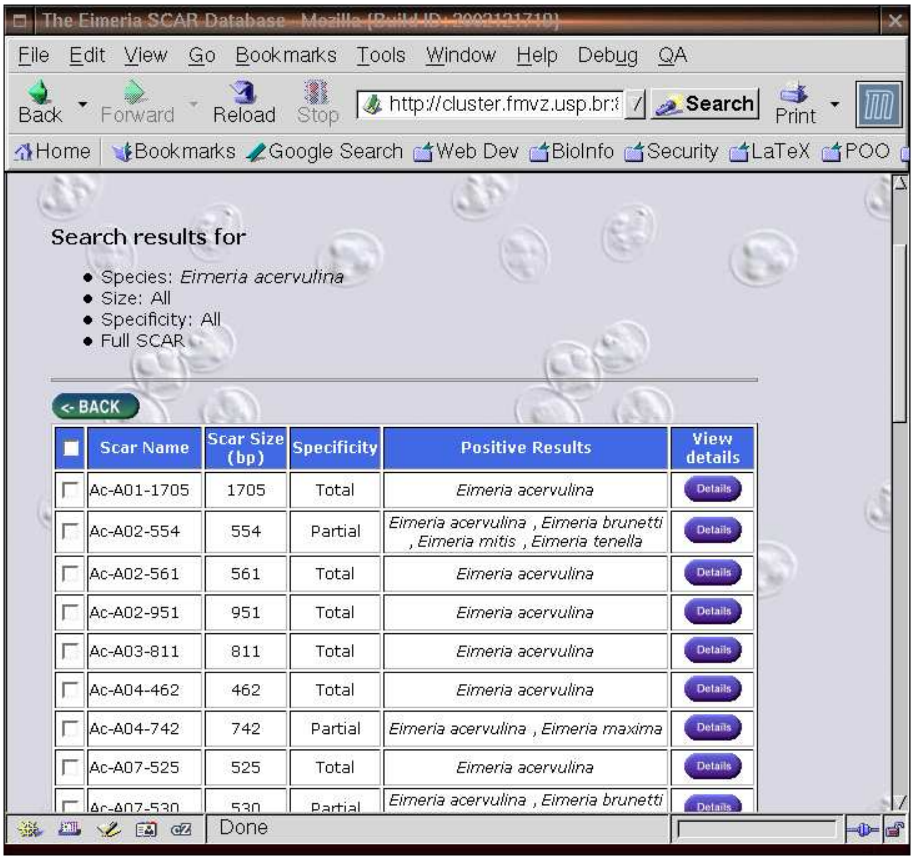

Figura 9: Buscando por SCARs, resultado de uma possível busca

Ao clicar em *Details*, encontramos mais informações sobre o SCAR selecionado. figura 10.

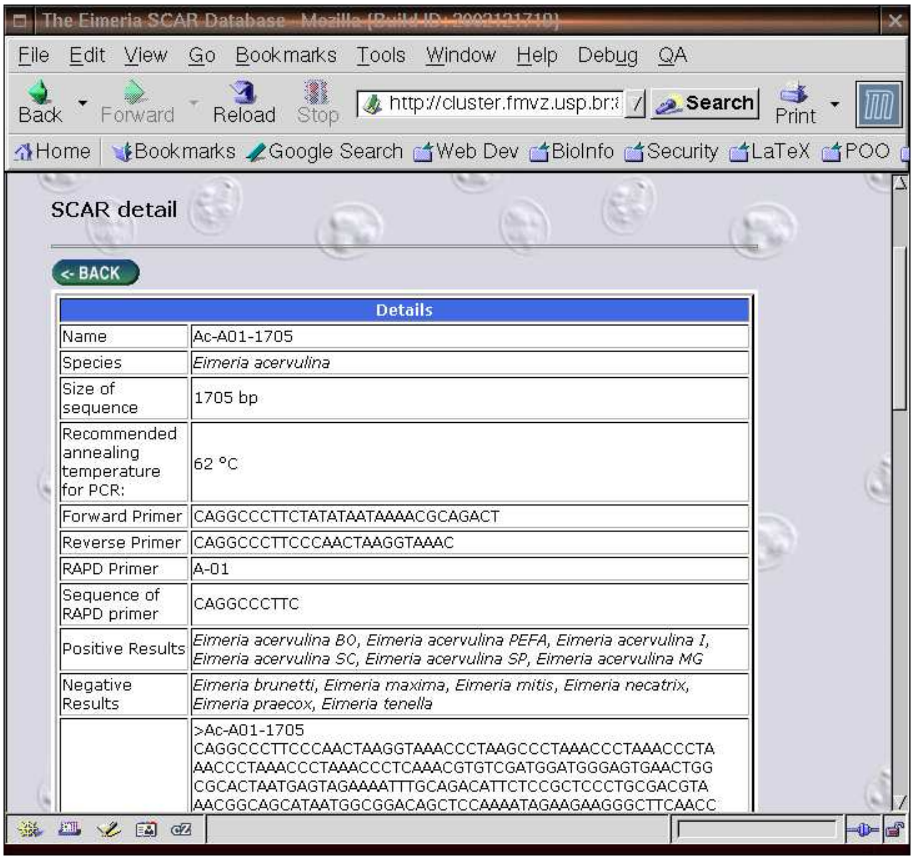

Figura 10: Mostrando os detalhes do SCAR, parte 1

Ao baixar um pouco o *scrollbar* do *Browser* observarmos uma imagem parecida com a figura 11. Nela temos *links* que apontam para resultados do *blastall*, assim como *links* para fazermos o *download* dos arquivos FASTA e dos cromatogramas das leituras de DNA correspondentes a esse SCAR.

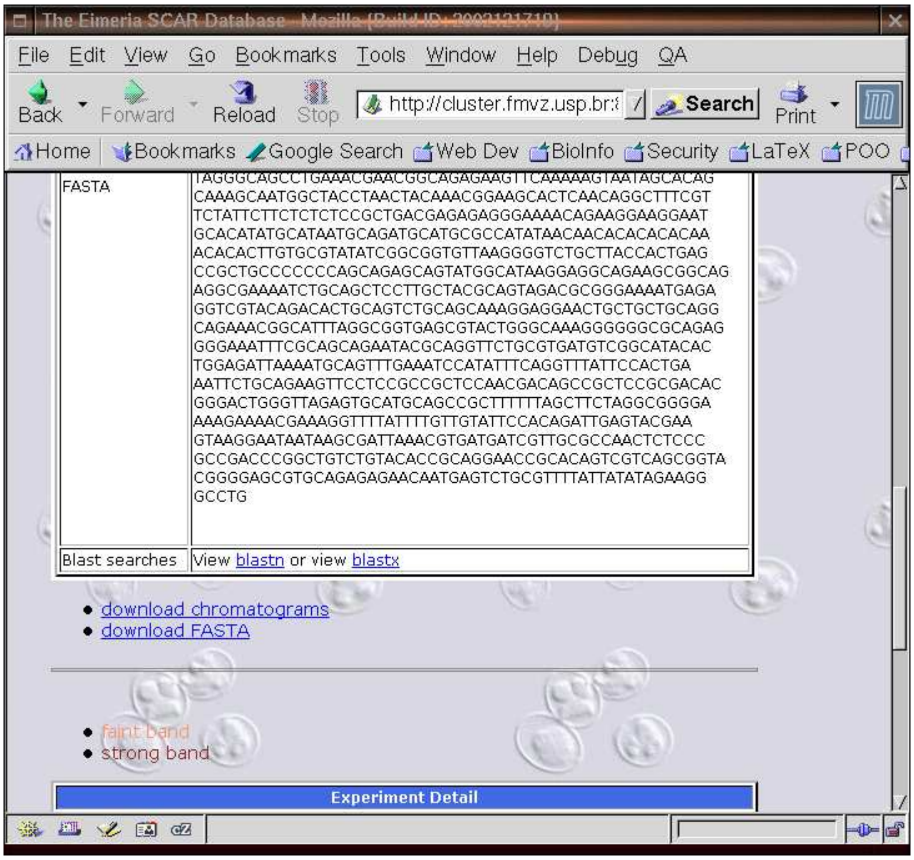

Figura 11: Mostrando os detalhes do SCAR, parte 2

Um pouco mais em baixo, podemos obter os resultados dos experimentos para verificar a especificidade desse SCAR.

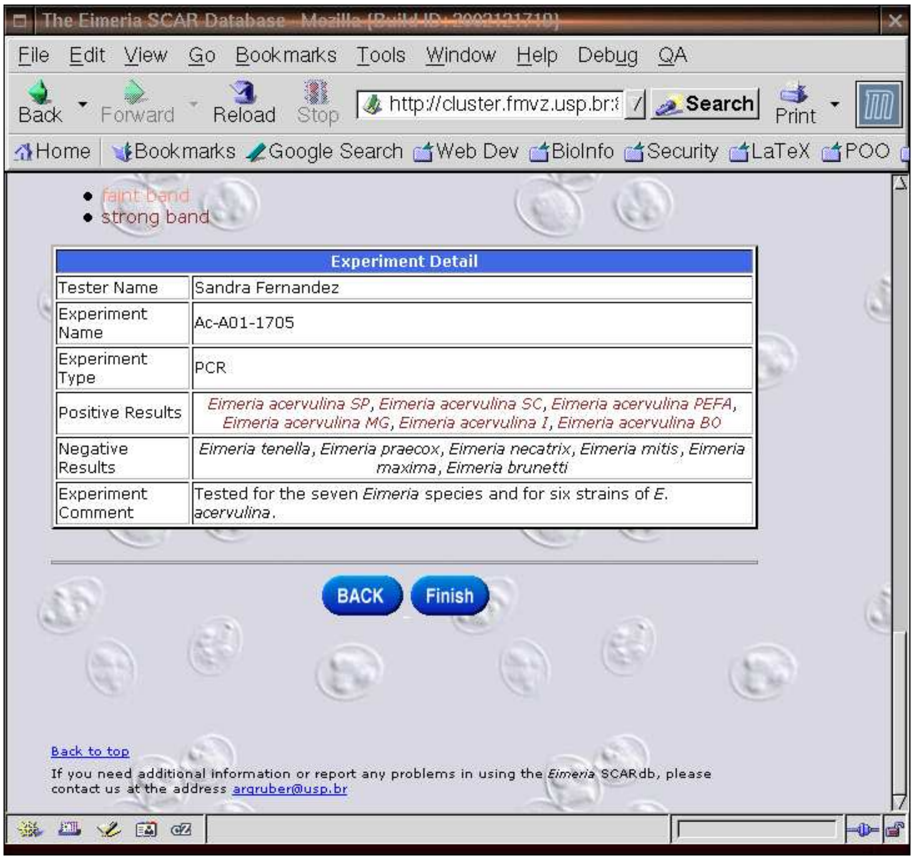

Figura 12: Mostrando o detalhes do SCAR, parte 3

4.3 SequenceObjectDB

A criação do SequenceObjectDB foi possível graças ao encapsulamento SequenceObject, os dois componentes podem ser trocados sem afetar o sistema de *pipelines*.

O SequenceObjectDB diminui o fluxo de dados que passa pelo pipeline. Ao executar o pipeline usando o SequenceObject, um processo recebe da entrada padrão um arquivo PHD, executa uma porção de operações, anexa os resultados no PHD e escreve o PHD modificado na saída padrão. No SequenceObjectDB, um processo coloca os resultados no banco, e passa somente ID da seqüência para frente.

Na execução de um pipeline que utiliza o SequenceObjectDB, houve uma grande perda de velocidade de execução. O motivo encontrado para justificar esta perda de perfomance foi que os dados não ficam na memória RAM e sim no disco.

Tentamos criar um *cache* para armazenar as informações mais requisitadas, e assim, melhorar esta situação. Isso não adiantou muito, já que a maior parte dos comandos SQLs são para atualizar as informações que estão em disco.

O problema de perfomance é o preço que pagamos para melhorar a forma em que os resultados são armazenados. Antes, era necessário uma estrutura complexa de diretórios, o que dificulta a análise geral de um projeto de seqüênciamento. Com o banco de dados, podemos criar um conjunto de consultas que geram relatórios globais e assim melhorar os resultados do seqüênciamento.

5 Considerações Futuras

O uso de padrões de projeto na implementação do Co-Ed, permite desenvolver novas *features* facilmente. Futuramente, podemos acrescentar novas características como o comando *Undo/Redo* cuja implementação utiliza os padrões Command e Memento.

Uma interface Web para acessar o banco EGeneDB é necessário. Já que pretendemos armazenar cerca de 19000 seqüência para o projeto de seqüênciamento de *Eimeira* spp. A interface fornecerá uma forma de disponibilizar os dados para outras pesquisas.

Atualmente a interface Web do SCARdb permite somente a inclusão de dados de *Eimeria* spp. Futuramente, essa interface deve permiter a inclusão de marcadores para outros seres.

Referências Bibliográficas

[1] Jorge Francisco del Teglia. Umlwp - uma ferramenta de edição de modelos de software baseados em uml. `http://www.linux.ime.usp.br/~cef/mac499-01/monografias/jorgefdt/`.

[2] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1994.

[3] David M. Geary. *JavaServer Pages avançado.* Sun microsystems, 2001.

[4] Arthur Grubber, Paulo H. Ahagon, Fernando T. G. Matsunaga, Alan M. Durham, Flavia Rainone, and André Y. Kashiwabara. E-gene - a modular and configurable pipeline system for automatic dna sequence analysis. 2003.

[5] DOE Human Genome Program. *Primer on Molecular Genetics.* 1992.

6 *Segunda Parte da monografia*

6.1 Desafios e frustrações encontrados

- Aprender os significados de alguns termos que os Biólogos falam: PCR, cepas, RAPD, SCAR, Gel, primer, vetor, quimera, cDNA, tRNA, mRNA, bibliotecas de ESTs, e muitas outras.
- Aprender algumas tecnologias que são muito usadas. JavaServer Pages, J2EE, CSS, JavaScript, Tomcat, PostgreSQL. Em apenas um semestre.
- Instalar os programas no sistema Tru64 Unix. O GCC instalado frequentemente me fornece um *Segmentation Fault*.
- Fazer pesquisas sobre predição de genes, neste caso não consegui entender os *papers* por falta de maturidade no assunto.

6.2 Lista das disciplinas cursadas no BCC mais relevantes

Muitas disciplinas foram importantes para o bom andamento do projeto:

- MAC0252 - Laboratório de Programação II. Esta disciplina me ofereceu um bom conhecimento sobre as linguagens Java e Perl.
- MAC0426 - Sistema de Bancos de Dados. Muito importante, pois desenvolvi dois bancos de dados.

- PCS0210- Redes de Computadores. Muitas vezes tive que configurar a rede do laboratório.
- MAC0441 - Programação Orientada a Objetos. Disciplina muito importante para aprender a programar direito.
- MAC0439 - Laboratório de Bancos de dados. Muito legal para aprender os passos que estão presentes durante a implementação de uma base de dados.

6.3 Interação com membros da equipe que tenham agido como mentores do trabalho

A equipe do laboratório sempre está presente para me ensinar os conceitos da Biologia Molecular. A aluna de doutorado Sandra Fernandez me ensinou os conceitos por traz da base de dados SCARdb. O estagiário Paulo Ahagon me ensinou sobre o sistema de pipelines. O aluno de iniciação Tiago Jose Paschoal Sobreira me ajudou a enfrentar as frustrações.

Eu devo agradecer ao Prof. Dr. Arthur Gruber que me incentivou nos momentos de dificuldade, o que serviu para agilizar o processo de desenvolvimento dos projetos. Ele sempre esta me informando sobre a Bioinformática.

O Professor Alan sempre está presente para eu eliminar dúvidas sobre computação, reuniões semanais servem para definir os próximos passos de cada projeto que estamos desenvolvendo. Ele sempre está disposto a corrigir todos os meus textos.

6.4 Diferenças notadas entre a forma de cooperação com colegas do BCC e a forma de trabalho conjunto no laboratório

No ponto de vista de desenvolvimento de software, os colegas do BCC são os meus consultores e os Biólogos são os meus clientes. Assim, os colegas do BCC discutem sobre as arquiteturas de softwares, as novas tecnólogias, as soluções de problemas. No laboratório, eu devo aplicar estes conhecimentos e aprender novos assuntos com as pessoas do laboratório para conseguirmos desenvolver um bom *software* de Bioinformática.

6.5 Observações sobre a aplicação de conceitos estudados nos cursos no contexto prático de aplicações reais

Novos conhecimentos foram facilmente entendidos graças ao estudo de algumas matérias ditas como "teóricas". O poder de abstrair foi obtida através das disciplinas que contém álgebra, a facilitade de aprender novas linguagens de programação foi obtida através do estudo da disciplina "Conceitos de Linguagem de Programação". A noção da quantidade de tempo gasto pelos algoritmos que escrevo foi obtida pela "Análise de Algoritmos".

6.6 Se o aluno fosse continuar atuando na área em que exerceu a iniciação, que passos tomaria para aprimorar os conhecimentos técnicos/metodológicos/científicos/etc relevates para esta atividade ?

O primeiro passo seria pesquisar sobre os assuntos relacionados à Bioinformática como algumas disciplinas da área de Ciências Biólogicas, Biologia Molecular, Bioquímica, Genética, etc. A participação em congressos será muito importante para trocar experiência e auxiliar no desenvolvimento de novas tecnologias.