4
Implementação

Este capítulo apresenta a implementação em Java de uma extensão para o BioSide, cujo objetivo é receber os dados de uma execução e armazenar em um banco de dados utilizando o esquema proposto no capítulo anterior. São apresentados também dois estudos de caso implementados no BioSide.

4.1.
Modelagem

Foram construídas as classes (Figura 22) ProvenanceReceiver e Persistence. A classe ProvenanceReceiver é usada para receber um objeto da classe Result, já disponível no BioSide para representar todas as informações relativas a execução de um workflow, inclusive a definição correspondente. A classe Persistence é usada para a gravação dos dados em banco, para a qual utilizamos o SGBD Postgresql.

A classe ProvenanceReceiver estende a classe StorageListenerAdapter, que é implementada no BioSide como um observador.

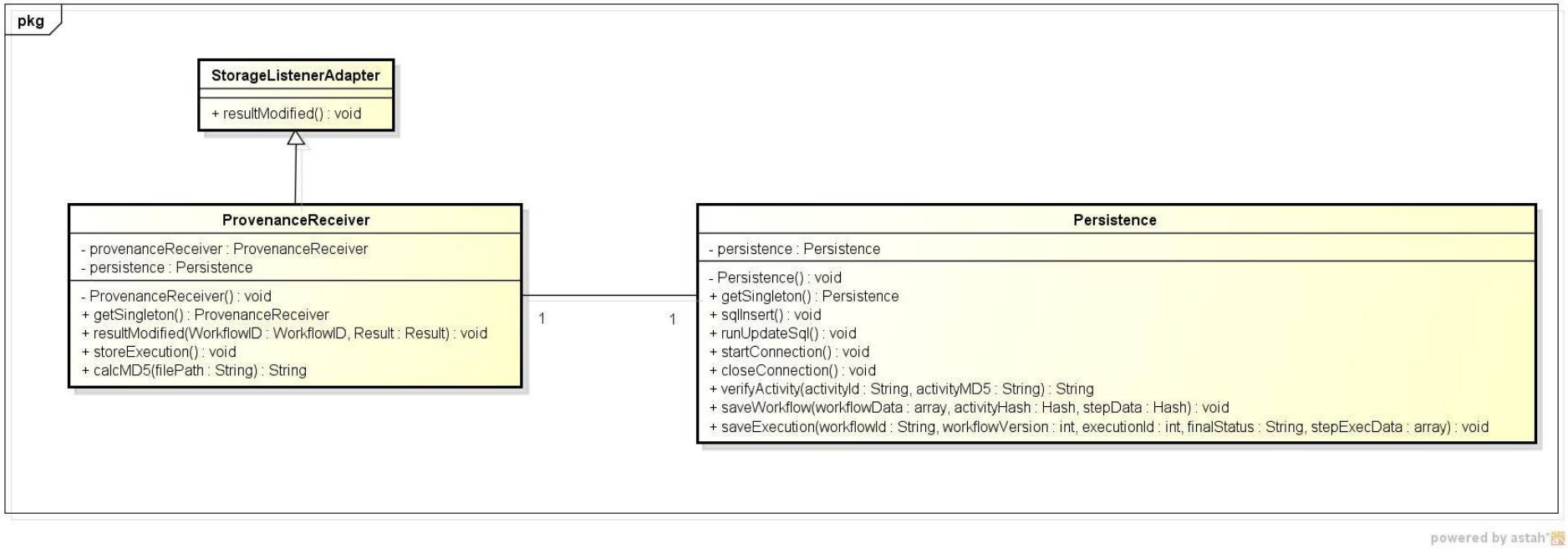

Figura 22 – Diagrama de Classes

A Figura 23 exibe o diagrama de sequência, incluindo troca de mensagens com algumas classes já existentes anteriormente no BioSide. No término da execução de um workflow, um evento é lançado e capturado pela classe ProvenanceReceiver através do método resultModified().

O método storeExecution() então é invocado, sendo responsável por ler os dados de execução, a definição correspondente ao workflow executado, as descrições das atividades, e gravar todas essas informações, invocando para tal a classe Persistence através de seu singleton.

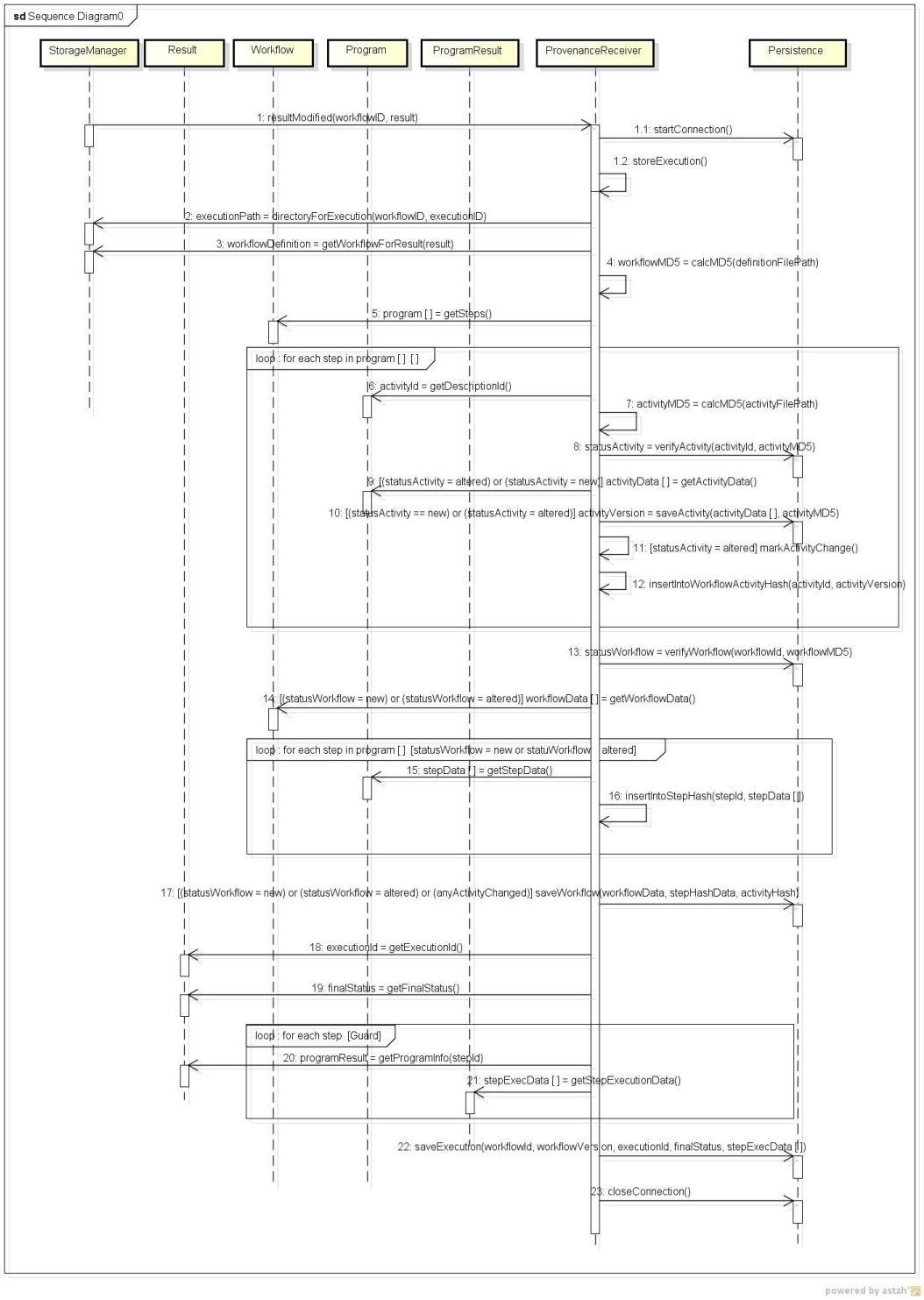

Figura 23 – Diagrama de Sequência

4.2.
Estudos de Caso

4.2.1.
Geração de Árvore Filogenética

Neste estudo de caso, apresentaremos um workflow (Figura 24) que tem por objetivo construir árvores filogenéticas a partir de sequências de proteínas. Este workflow foi obtido da seção de exemplos do site do sistema BioSide.

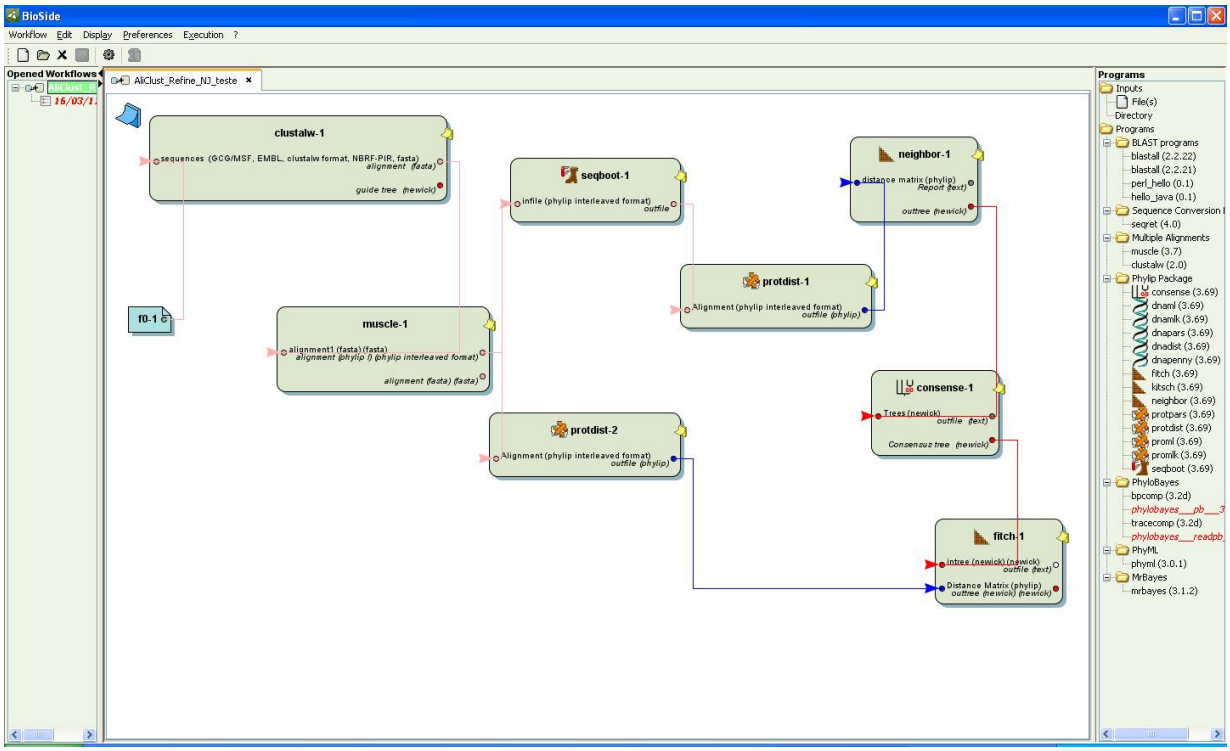

Figura 24 – Workflow para construção de árvore filogenética

Os passos principais para a criação de uma árvore filogenética são:

1. Identificar uma sequência de proteína (ou DNA) de interesse
2. Identificar outras sequências relacionadas com a sequência de interesse
3. Alinhar as sequências
4. Usar o alinhamento resultante para gerar a árvore filogenética

Para executar cada um desses passos existem diferentes métodos e inúmeros programas disponíveis na web. O presente workflow utiliza os programas ClustalW [Clustalw2, 2011] e muscle [Muscle, 2011] para o

alinhamento de sequências e programas disponibilizados no pacote PHYLIP [Phylip, 2011] para a geração da árvore filogenética. Todos esses programas podem ser acessados pelo BioSide uma vez que o sistema possui atividades que descrevem cada um deles.

Iremos descrever brevemente os passos deste workflow.

Para o alinhamento de sequências, o workflow utliza o programa clustalw para um primeiro alinhamento múltiplo entre as sequências de proteínas GST e posteriormente o programa muscle para refinar o alinhamento do clustalw.

Para gerar a árvore filogenética, o workflow fornece dois procedimentos independentes, utilizando dois métodos distintos do pacote PHYLIP, que lidam com dados na forma de matrizes com distâncias entre pares de sequências.

Para construir uma filogenia no PHYLIP, usando um método de distância, é necessário primeiro executar um programa para calcular uma matriz de distância entre as sequências do alinhamento (o workflow utiliza o PROTDIST) para então utilizar um programa para a construção da árvore a partir dessa matriz (o workflow utiliza o NEIGHBOR e o FITCH).

Utilizando o NEIGHBOR, que calcula árvores pelo método Neighbor-Joining, a saída do muscle é usada como entrada para o seqboot, que realiza uma amostragem através do método de bootstrap, e o resultado é um arquivo com múltiplos alinhamentos gerados a partir do alinhamento original. A saída do seqboot é submetida para o programa protdist que calcula matrizes de distância para cada um dos conjuntos de dados, gerando um arquivo de saída que será utilizado no programa neighbor para a construção de múltiplas árvores. O arquivo de saída do neighbor é submetido ao programa consense que avalia a significância das análises encontrando o consenso das árvores geradas para cada amostra.

O FITCH calcula árvores pelo método de Fitch-Margoliash, least-squares de Cavalli-Sforza e Edwards (ou outro método similar da mesma família). São métodos mais lentos e rigorosos e, portanto, muitas vezes o uso de bootstrap pode ser impeditivo. Nesse caso, no workflow, o alinhamento múltiplo original do MUSCLE serve de input direto para a construção da matriz de distância pelo PROTDIST, que servirá de input para o programa FITCH que gera uma única melhor árvore final.

Os outputs do NEIGHBOR e do FITCH são arquivos que contêm diagrama(s) de árvore(s) e descrições de árvore(s) no formato Newickian.

Este workflow utiliza apenas programas de linha de comando e como entrada um arquivo fornecido pelo usuário, que são características interessantes para o escopo deste trabalho.

O estudo de caso de Geração de Árvore Filogenética já estava implementado no BioSide como um workflow de exemplo disponível para download.

4.2.2. MHOLline

Descreveremos agora com mais detalhe a aplicação MHOLline, apresentando sua implementação atual e a que foi realizada no SGWC BioSide.

4.2.2.1. Implementação Atual

A implementação atual do workflow consiste em três componentes:

- Aplicação Web
- Banco de dados mholdb
- Scripts Perl

A aplicação web foi construída em PHP para que os usuários possam executar o workflow MHOLline via web. O sistema permite uma execução limitada para usuários não cadastrados, que podem submeter um arquivo de entrada com no máximo 50 sequências de aminoácidos. Para estes usuários, os resultados ficam disponíveis para download durante 15 dias, sendo totalmente apagados ao fim deste período. Para usuários cadastrados, o sistema permite o envio de arquivos de seqüências de quaisquer tamanhos, e nunca apaga os resultados, mantendo um histórico de todas as execuções feitas pelo usuário.

No banco de dados mholdb, para o qual se utiliza o SGBD MySQL, algumas informações sobre os processos de cada usuário são gravadas. Os resultados dos programas BATS, FILTERS, e ECNGET também são registrados em tabelas do banco.

O MHOLline grava algumas saídas de programas em um banco de dados relacional. O programa BATS por exemplo grava em banco os alinhamentos gerados pelo BLAST com algumas outras informações, por exemplo o grupo no qual se enquadra o alinhamento.

Já os scripts representam o workflow científico propriamente dito. A aplicação web invoca um script de inicialização, que insere o processo do

usuário no banco de dados e dispara a execução do script principal do workflow, responsável pela execução em sequência dos programas necessários e eventuais tratamentos de dados.

4.2.2.2. Implementação no BioSide

A implementação do workflow MHOLline no BioSide necessitou da transformação do script principal em vários scripts que representam cada um as atividades utilizadas no workflow. Para fins de simplificação desconsideramos os programas HMMTop, Ecnget e Procheck.

A Figura 25 mostra de maneira simplificada a arquitetura original do MHOLline que foi considerada neste trabalho. A aplicação web invoca o script upload.pl, que grava o arquivo fornecido pelo usuário, registra o processo no banco mholdb e gera um *uploadId*, que representa a solicitação de serviço. Ao final o script gera um arquivo *exec.sh*, responsável apenas por invocar o script principal. Um processo chamado *execManager* fica em constante execução para invocar os scripts exec.sh, e assim executar finalmente o workflow mholline. Apenas o script principal (mholline.pl) foi considerado neste trabalho e inserido como um workflow no BioSide. Dessa forma o script exec.sh passa a invocar o BioSide (Figura 26) ao invés de invocar o script mholline.pl. A Figura 27 mostra o workflow no BioSide.

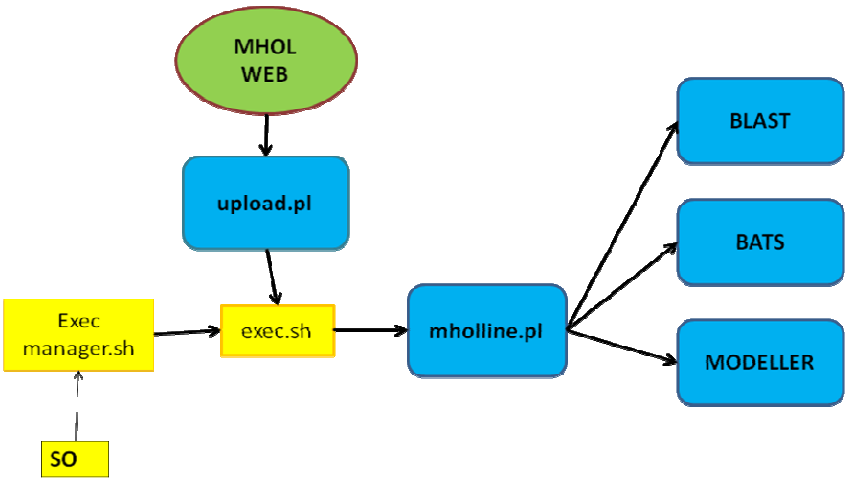

Figura 25 – Arquitetura original do MHOLline

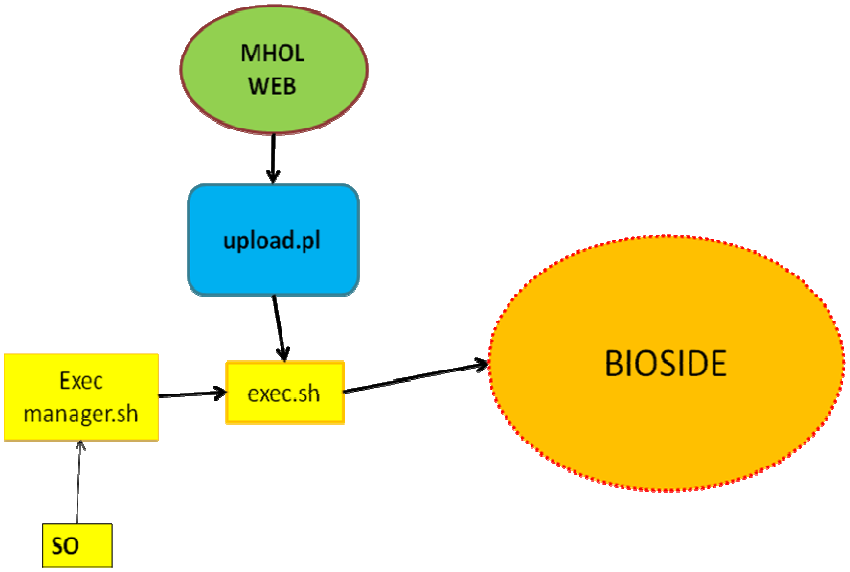

Figura 26 – MHOLline com chamada ao BioSide

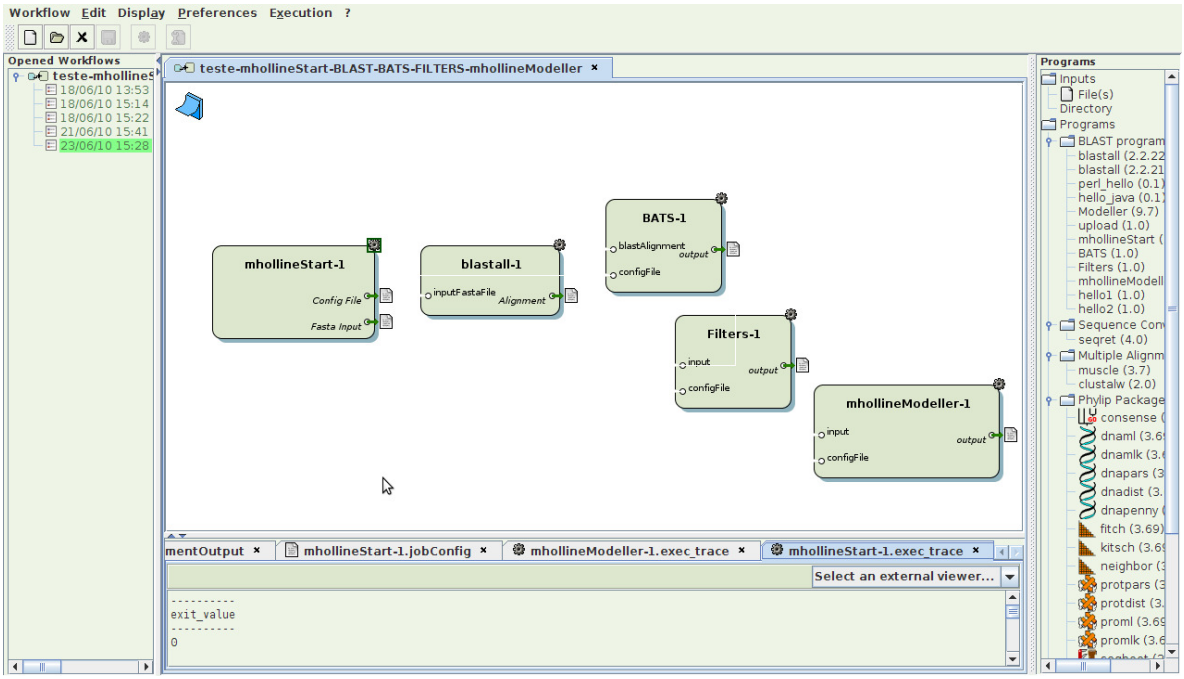

Figura 27 – Workflow MHOLline implementado no BioSide

Algumas alterações tiveram que ser realizadas nos scripts para permitir a modelagem do workflow de acordo com o esquema proposto.

No início do workflow, o arquivo de sequências fornecido é dividido em vários arquivos menores. Cada arquivo é submetido ao BLAST, gerando vários resultados. O programa BATS é então executado e cada relatório gerado pelo BLAST é avaliado, gerando ao final a classificação das sequências em grupos e

a escolha de um template campeão para cada sequência de entrada. Esse processamento é realizado em uma estrutura de iteração, para cada arquivo gerado após o processamento inicial do arquivo de entrada.

A segunda estrutura de iteração ocorre na chamada do programa Modeller, na qual para cada sequência de entrada será elaborado o modelo final.

Para permitir a modelagem do workflow como um data-flow sem iteração retiramos a quebra inicial do arquivo, submetendo-o inteiro ao BLAST. A segunda iteração foi encapsulada dentro da atividade mhollineModeller, que prepara os arquivos e invoca o programa Modeller.

Além das iterações, temos fluxos de controle. O workflow original é um data-flow apenas até o BATS, que grava no banco mholdb os dados resultantes de seu processamento, a serem lidos pelos próximos programas. Após o BATS os programas serão executados em sequência, sem passagem direta de dados entre os mesmos. Para transformar o workflow em um data-flow passível de implementação no BioSide e representação em nosso projeto, alteramos os scripts para gerar um arquivo vazio, apenas para servir de entrada para o próximo programa.

Com essas alterações foi possível modelar e executar o script principal como um workflow no BioSide.

4.3. Exemplos Práticos de Contribuições

Na seção 3.2 foi descrito brevemente como alguns dos desafios estudados passaram a ser suportados através do esquema conceitual proposto e implementado em banco de dados relacional. Para fins de simulação, foram criadas algumas instâncias de dados como exemplos para o melhor entendimento do esquema em relação aos desafios abordados anteriormente. As figuras 28 a 39 apresentam instâncias relativas a execuções dos workflows de Geração de Árvore Filogenética e MHOLline. Omitimos os arquivos de definição para facilitar a visualização das instâncias. No anexo 2 apresentamos dois exemplos desses arquivos.

workflowId	workflowVersion	workflowDef	workflowName	workflowDefHash
1	1	<arquivo descritor do workflow>	ArvoreFilogenetica	7d9330d...c84e6ec
1	2	<arquivo descritor do workflow>	ArvoreFilogenetica	cb46a24...58fd67d
2	1	<arquivo descritor do workflow>	MHOLlineMain	e95ddb2...a88971
3	1	<arquivo descritor do workflow>	MHOLlineRefine	0406213...dd3f3b
4	1	<arquivo descritor do workflow>	MHOLlineTest	e51993c...211d9

Figura 28 – Tabela Workflow

stepId	workflowId	workflowVersion	activityId	activityVersion
Clustalw_1	1	1	Clustalw	1
Clustalw_1	1	2	Clustalw	1
Muscle_1	1	1	Muscle	1
Muscle_1	1	2	Muscle	2
BATS_1	2	1	BATS	1
ChooseSeq_1	3	1	ChooseSeq	1
BATS_1	4	1	BATS	2

Figura 29 – Tabela Step

activityId	activityVersion	activityDescription	activityCode	activityHash	programId
clustalw	1	Atividade de chamada ao programa Clustalw	<arquivo descritor da atividade clustalw>	171d0997b4bd10d9189e3a90faad9524	1
Muscle	1	Atividade de chamada ao programa Muscle	<arquivo descritor da atividade Muscle>	a460637cf1b761eb44b6e23bc8b971e3	2
Muscle	2	Atividade de chamada ao programa Muscle	<arquivo descritor da atividade Muscle>	a460637cf1b761eb44b6e23bc8b971e3	3
BATS	1	Atividade de chamada ao script BATS	<arquivo descritor da atividade BATS>	dfafc7ca6339d9d2ec73d8f4912b72e8	15
ChooseSeq	1	Atividade para escolha de outras sequências pdb.	<arquivo descritor da atividade ChooseSeq>	334196cb0ea1f81dfb7f2b943ac671ed	16
BATS	2	Atividade de chamada ao script BATS	<arquivo descritor da atividade BATS>	90bf674375fa29f7c5c2681a248c4577	15

Figura 30 – Tabela Activity

programId	programName	programProvider	programVersion	programType
1	clustalw.exe	http://www.ebi.ac.uk/Tools/msa/clustalw2/	2.0.11	Alinhamento múltiplo de sequências
2	Muscle.exe	http://www.drive5.com/	3.6	Refinamento de alinhamento
3	Muscle.exe	http://www.drive5.com/	3.7	Refinamento de alinhamento
14	BLAST	NCBI	2.2.22	Alinhamento de Sequências
15	BATS	User	1.0	Pontuação e escolha de sequências do PDB com base no alinhamento feito pelo BLAST
16	ChooseSeq	User	1.0	Escolha do usuário por outras sequências do PDB com base na pontuação feita pelo BATS.

Figura 31 – Tabela ExternalResource

parameterName	parameterRole	activityId	activityVersion
operationType	input	clustalw	1
sequences	input	clustalw	1
alignment	output	clustalw	1
Alignment1	input	Muscle	1
score_classification	input	ChooseSeq	1

Figura 32 – Tabela Parameter

portLinkId	from_stepId	from_workflowId	from_workflowVersion
1	Clustalw_1	1	1
2	Clustalw_1	1	2

to_stepId	to_workflowId	to_workflowVersion
Muscle_1	1	1
Muscle_1	1	2

from_activityId	from_activityVersion	from_parameterName
clustalw	1	alignment
clustalw	1	alignment

to_activityId	to_activityVersion	to_parameterName
Muscle	1	alignment1
Muscle	2	alignment1

Figura 33 – Tabela PortLink

opmGraphId	workflowId	workflowVersion
1	1	1
2	1	2
11	2	1
12	3	1

Figura 34 – Tabela OPMGraph

opmProcessId	started	finished	stepId	workflowId	workflowVersion	opmGraphId
1	2011-04-26 18:10:30	2011-04-26 18:11:13	Clustalw_1	1	1	1
2	2011-04-26 18:11:15	2011-04-26 18:13:07	Muscle_1	1	1	1
35	2011-04-26 17:19:09	2011-04-26 17:22:00	BATS_1	2	1	11
36	2011-04-30 10:02:09	2011-04-30 10:02:12	ChooseSeq _1	3	1	12

Figura 35 – Tabela OPMProcess

opmArtifactId	artifactValue	artifactShortValue	artifactMD5	artifactPath
1	null	>1EOG.pdb PYTVVYFPVRGRCA ALRMLLADQGQSW KEEVVTVETWQ	87b98aaf0e0 5aa17e5695 474f050bc8b	C:/program files/bioside/user/priscila/ ArvFilo/2011-04-26T18_10_27.250+02000/seqprot.fasta
2	multiple sequences alignment	null	7b4dd1e786 6aecd9de6df 16e37ac745 b	null
112	null	1VS7 14.3000 2WH1 16.5000 1JGO 14.4000	05f94cebbac 28299d5f794 2aa9af3bf8	C:/program files/bioside/user/priscila /MHOLlineMain/2010-09-24T10_36_10.250+02000/bats_results

opmProcessId	role
1	input
1	input
35	output

Figura 36 – Tabela OPMArtifact

parameterValueId	parameterValue	parameterName	activityId	activityVersion
1	C:/bases/seqprot.fasta	sequences	clustalw	1
2	multiple sequences alignment	operationType	clustalw	1
11	null	score_classification	ChooseSeq	1

stepId	workflowId	workflowVersion
Clustalw_1	1	1
Clustalw_1	1	1
ChooseSeq_1	3	1

Figura 37 – Tabela InputValue

parameterValueId	artifactId
11	112

Figura 38 – Tabela InputValue_Artifact

annotationId	property	value	opmProcessId
1	md5	eb48a6c26fa37afbda4b00b13c9f2732	2
2	path	C:\programasBioInfo\Muscle.exe	2

Figura 39 – Tabela Annotation

4.3.1. Reprodutibilidade

Nas subseções seguintes discutiremos com exemplos como foram atingidos os objetivos de reutilizar uma definição de workflow e reproduzir estritamente uma execução.

4.3.1.1. Reuso de Definição

As tabelas Workflow, Step, Activity, ExternalResource, Parameter, InputValue, PortLink, BioInput e PrimitiveParameter permitem a descrição de uma especificação de workflow. Observamos que a tabela Workflow (Figura 28) registra duas versões do workflow do estudo de caso Geração de Árvore Filogenética. Na tabela Step (Figura 29) temos alguns dos passos deste workflow, Clustalw_1 e Muscle_1, que por sua vez fazem referência às respectivas atividades invocadas (clustalw e Muscle). A tabela PortLink (Figura 33) registra as ligações entre os passos, por exemplo a ligação entre os passos Clustalw_1 e Muscle_1, exibida na tabela. Toda a definição referente a uma execução fica portanto registrada de maneira permanente nestas tabelas.

Suponha a execução registrada em OPMGraph (Figura 34, OPMGraphId = 1). Esta execução possui uma referência para a definição de workflow que foi utilizada na execução, no caso o workflow de Geração de Árvore Filogenética na versão 1. Caso o usuário altere a definição deste workflow e o execute novamente, o módulo de proveniência ao capturar esta execução irá criar uma nova versão para a referida definição. Por exemplo, se o usuário alterou a versão do programa Muscle, passando a utilizar a versão 3.7 substituindo a 3.6, será criada a versão 2 da definição do workflow, o que significa que todas as instâncias relativas a esta definição receberão uma nova versão. Este versionamento efetuado pelo módulo de proveniência permite que o usuário tenha acesso a especificação do workflow tal como existia no momento da execução. A alteração na definição do workflow não sobrescreve a existente anteriormente (versão 1), cujos registros permanecerão intactos para que o usuário possa consultá-los e se necessário reutilizá-los.

4.3.1.2.
Reprodutibilidade Estrita

Na seção 2.2.3.2 ressaltamos a importância do registro dos dados e programas para permitir a reprodução estrita de uma execução. O caminho completo dos programas executados é gravado pela extensão de proveniência na tabela Annotation, como anotação referente ao processo executado. Também é registrado como anotação o valor de *hash* md5 do arquivo do programa. Nas tabelas Step e OPMProcess (Figura 29 e Figura 35), vemos que o passo Muscle_1 do workflow de geração de árvore filogenética foi executado, gerando um processo (opmProcessId = 2). Na tabela Annotation (Figura 39) foram registrados o *path* completo e o md5 do programa. O valor de *hash* permite que o sistema verifique se algum programa utilizado no workflow foi alterado e alerte o usuário caso este deseje reproduzir estritamente a execução.

Além desses registros em relação aos programas executados é possível também registrar dados e metadados sobre os dados consumidos e produzidos em uma execução. Na tabela OPMArtifact (Figura 36), observamos que foram registrados alguns metadados sobre o dado consumido pelo processo relativo ao passo Clustalw_1. Este dado (OPMArtifactId = 1) é o arquivo *seqprot.fasta*, para o qual temos o shortValue do arquivo, que é uma extração da parte inicial do mesmo, o valor de *hash* md5, e o caminho completo.

4.3.2.
Gerência de Dados Consumidos e Produzidos

Todos os dados gerados e consumidos são registrados como artefatos na tabela OPMArtifact, sendo que arquivos de dados serão copiados para uma pasta do sistema (seção 3.2.2).

No workflow de árvore filogenética temos na tabela OPMArtifact (Figura 36) dois registros de artefatos, um que se refere ao parâmetro de configuração do programa Clustalw, e outro que se refere ao arquivo que o mesmo programa consome. Vemos nas tabelas Activity e Parameter (Figura 30 e Figura 32) que a atividade Clustalw consome um parâmetro de configuração chamado operationType, que define qual tipo de operação deve ser executada pelo programa clustalw. Na definição do workflow o valor desse parâmetro é “multiple sequences alignment”, e esse registro fica na tabela InputValue (Figura 37). Ao ser executado, um artefato referente a esse parâmetro é gerado na tabela OPMArtifact. O outro artefato presente na tabela OPMArtifact (Figura 36) se

refere ao arquivo de sequências fornecido como entrada para o passo clustalw_1. São registrados o md5 do arquivo, o início do arquivo para facilitar a identificação do mesmo, e o caminho completo. O arquivo pode ser copiado para uma pasta do sistema conforme explicado na seção 3.2.2, e na tabela OPMArtifact são registrados esses metadados sobre o mesmo.

4.3.3. Reuso de Dados

Nas instâncias de execução do workflow MHOLline, temos um caso de reuso de dados mantendo proveniência. Temos duas definições de workflow, o workflow Principal e o workflow de Refinamento. O workflow principal (Figura 28, registro MHOLlineMain) possui um passo responsável por invocar a atividade BATS. Em alguma execução deste workflow, o passo BATS_1 foi executado, tendo esta execução sido registrada como um OPMProcess, (registro opmProcessId = 35). O referido processo produziu um artefato (Figura 36, registro opmArtifactId = 112), a saber um arquivo com as pontuações que o programa BATS encontrou para cada sequência do banco de dados PDB. No workflow de refinamento (registro workflowId = 3), temos o passo ChooseSeq_1, que irá executar a atividade de escolha de outra (s) sequência (s) para a modelagem tridimensional. Esta atividade possui um parâmetro de entrada chamado score_classification, que é uma classificação de pontuações tal como a produzida no passo BATS_1 do workflow principal. Caso o usuário deseje utilizar o artefato já produzido (id = 112) como entrada do passo ChooseSeq_1, esta escolha ficará registrada no relacionamento InputValue_Artifact (Figura 38, registro parameterValueId = 11 e artifactId = 112).

Este exemplo demonstra como em uma especificação de workflow podem ser utilizados artefatos produzidos por execuções passadas, mantendo a rastreabilidade para a origem destes artefatos.

4.3.4. Descrição e Gerência de Atividades

Uma das novidades do modelo proposto em relação aos modelos estudados é a separação entre o contexto de descrição da atividade e do programa externo que é executado pela mesma. Temos por exemplo o registro da atividade BATS na tabela Activity (Figura 30), e do programa BATS na tabela ExternalResource (Figura 31).

Outra contribuição é registrar para um determinado passo de workflow não apenas um identificador da atividade, mas a atividade propriamente dita. O workflow MHOLlineMain possui um passo chamado BATS_1, que é uma instanciação da atividade BATS. O passo BATS_1 deve referenciar a atividade BATS, que por sua vez tem o seu código descritor armazenado na tabela Activity.

Suponha um cenário no qual o usuário efetue alguma alteração na atividade BATS, por exemplo, alterando o nome de um parâmetro. Após esta alteração, suponha que o usuário crie um novo workflow, chamado MHOLlineTest, e inclua um passo que invoca a atividade BATS, agora alterada. Ao executar este workflow, a alteração da atividade será identificada pela extensão de proveniência (através da comparação entre os códigos MD5) e uma nova versão para a mesma será criada, mantendo a versão anterior intacta.

Com relação aos recursos externos, na tabela ExternalResource (Figura 31) temos metadados sobre os programas utilizados nos estudos de caso. É possível registrar por exemplo que a versão do programa BLAST a ser utilizada no workflow MHOLline é a 2.2.22.

4.3.5. Modelo de Proveniência

As contribuições em relação a modelagem conceitual de proveniência foram listadas na seção 3.2.5. Aqui iremos dar alguns exemplos de consultas que podem ser realizadas a partir da implementação em abordagem relacional, utilizando o BioSide como exemplo.

O BioSide originalmente armazena as definições de workflows em arquivos XML, assim como informações sobre a execução dos mesmos. Todas as informações (especificação e execuções) sobre um determinado workflow são registradas em um diretório que recebe o nome que o usuário escolheu para o workflow. Este diretório contém a especificação do workflow em XML, e para cada execução realizada é criado um diretório para armazenar registros sobre a mesma, como arquivos utilizados e log de execução.

Esta organização em arquivos dificulta que sejam feitas consultas a proveniência, ou até mesmo impossibilita diversas consultas que podem ser realizadas utilizando a abordagem relacional proposta neste trabalho. Vejamos alguns exemplos de consultas que podem ser construídas utilizando SQL:

Consulta 1: Quais especificações de workflows utilizam a versão 2.2.18 do programa BLAST?

Consulta 2: Em quais versões do workflow MHOLline o parâmetro e-value do BLAST possui o valor 0,1?

Consulta 3: Em alguma execução do workflow de árvore filogenética foi produzida uma árvore, que se encontra no caminho "C:\OUTPUT\FILOGENETICA_2011_01_01.seq". Esta árvore foi reutilizada em outra definição de workflow?

A figura (Figura 40) exibe as consultas listadas traduzidas para SQL. Essas consultas e muitas outras podem ser formuladas, o que não é possível sem a extensão de proveniência implementada.

```sql
SELECT w.workflowname, w.workflowversion
FROM workflow w, step s, activity a, externalresource e
WHERE
    w.workflowid = s.workflowid and
    w.workflowversion = s.workflowversion and
    s.activityid = a.activityid and
    s.activityversion = a.activityversion and
    a.programid = e.aprogramid and
    e.programname = 'BLAST' and
    e.programversion = '2.2.18'
```

```sql
SELECT w.workflowname, w.workflowversion
FROM workflow w, step s, activity a, externalresource e,
     parameter p, inputvalue i
WHERE
    w.workflowid = s.workflowid and
    w.workflowversion = s.workflowversion and
    s.activityid = a.activityid and
    s.activityversion = a.activityversion and
    a.programid = e.aprogramid and
    a.activityid = p.activityid and
    a.activityversion = p.activityversion and
    p.activityid = i.activityid and
    p.activityversion = i.activityversion and
    p.parametername = i.parametername and
    w.workflowname = 'MHOLline' and
    e.programname = 'BLAST' and
    p.parametervalue = 'e-value' and
    i.parametervalue = '0,1'
```

```sql
SELECT w.workflowname, w.workflowversion
FROM workflow w, step s, inputvalue i, inputvalue_artifact ia,
     opmartifact a
WHERE
    w.workflowid = s.workflowid and
    w.workflowversion = s.workflowversion and
    s.workflowid = i.workflowid and
    s.workflowversion = i.workflowversion and
    s.stepid = i.stepid and
    i.parametervalueid = ia.parametervalueid and
    ia.opmartifactid = a.opmartifactid and
    a.artifactpath = 'C:\OUTPUT\FILOGENETICA_2011_01_01.SEQ'
```

Figura 40 – Consultas 1, 2 e 3 em SQL

4.4. Conclusão

Neste capítulo apresentamos uma extensão simples para o BioSide que captura dados de proveniência e os armazena em banco de dados. A implementação não foi feita de forma genérica, uma vez que o objetivo principal era demonstrar a utilização do projeto proposto em uma situação real.

Os estudos de caso foram representados e executados no BioSide utilizando a extensão de proveniência. Para representar o MHOLline algumas adaptações foram feitas a fim de eliminar iterações e fluxos de controle.

Foram listados também exemplos práticos de como os desafios são tratados de acordo com a modelagem proposta.