7 Prototipação do SGWBio

7.1 Introdução

Este capítulo apresenta um protótipo de SGWBio que contempla os seguintes módulos: controlador, assistente, gerente de ontologia, gerente de otimização, gerente de execução e compartilhamento de objetos.

A implementação, feita na linguagem Java, foi baseada na instanciação, para o ambiente pessoal, dos *hot spots* do *framework* esquematizado na Figura 9 para o ambiente pessoal. O módulo assistente é representado pela classe *DesktopAssistant* da Figura 12, o módulo gerente de ontologia é representado pela classe *AMZIPrologOntologyManager* da Figura 13, o módulo gerente de otimização é representado pela classe *PersonalOptimizationManager* da Figura 14, o módulo gerente de execução é representado pela classe *PersonalExecutionManager* da Figura 15, o módulo controlador é representado pela classe *Controller* da Figura 11 e o módulo compartilhamento de objetos é composto por várias classes incluindo as listadas na Figura 10.

As próximas seções descrevem aspectos da implementação, utilizando como ilustração o workflow do genoma completo apresentado na Figura 1.

7.2 Assistente

No protótipo, o módulo assistente foi implementado instanciando a classe *DesktopAssistant*, ilustrado na Figura 12.

A interface do sistema assistente, apresentada na Figura 31, é organizada como se segue. O painel à direita, chamado *Workflow View*, apresenta uma árvore do workflow corrente, ou seja, o que está sendo definido pelo pesquisador. O painel central, chamado de *Process View* exibe uma hierarquia de classes de processos. O painel à esquerda está dividido em quatro sub-painéis. O sub-painel do topo, chamado de *Project View*, apresenta as classes e instâncias de projeto. O segundo sub-painel, chamado de *Resource View*, mostra a hierarquia de classes e

instâncias de recursos. Os dois últimos sub-painéis, apresentam as hierarquias de classes e instâncias de dados de entrada e saída. Estes sub-painéis são chamados de *Input View* e *Output View*, respectivamente.

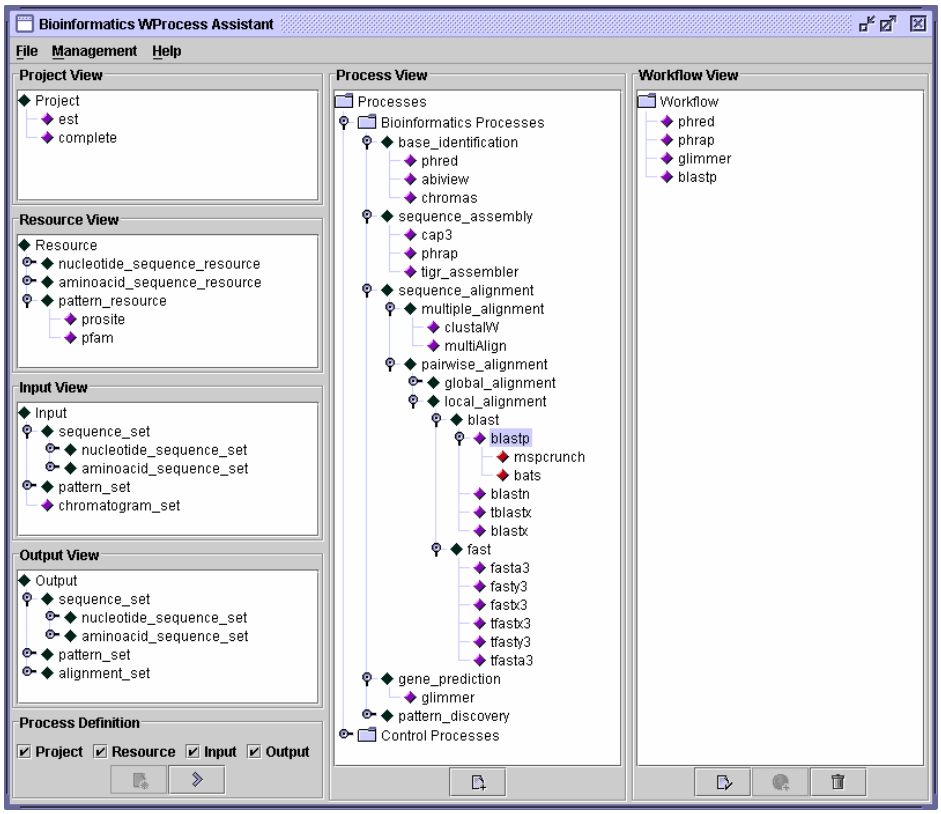

Figura 31. Inferface do módulo assistente no protótipo.

Definição de Instâncias de Processos de Bioinformática

No princípio da criação do workflow de genoma completo apresentado na Figura 1, o pesquisador está de posse dos cromatogramas e deseja identificar as bases dos *reads*. Neste caso ele pode selecionar diretamente uma das instâncias dos processos da classe *Base Identification* no painel central. Para incluir uma instância de processo no workflow corrente, o pesquisador seleciona a instância diretamente no *Process View*, seleciona a posição em que deseja incluí-la no *Workflow View* e clica no botão do *Process View* para realizar a inclusão.

Caso o pesquisador esteja em dúvida para escolher qual a classe de processos mais indicada para sua escolha, ele pode usar os sub-painéis da esquerda para restringir as suas possibilidades de escolha no *Process View*. Sendo assim, ele pode selecionar *Chromatogram Set* no painel *Input View*. O assistente

apresentará todos os processos que aceitam cromatogramas como entrada. Neste caso somente as classes de processos Phred e Chromas serão apresentadas no painel *Process View*.

Caso o pesquisador não saiba escolher entre as duas instâncias, ele pode clicar com o botão direito sobre o nome da classe no painel *Process View* e o assistente apresentará, em uma janela, as qualidades que diferenciam as classes Phred e Chromas. Por exemplo, o custo e a popularidade são qualidades que as diferenciam, pois o Phred é um software livre enquanto o Chromas precisa de licença. Nesta janela, o pesquisador deve definir pesos para cada uma das qualidades. Se o pesquisador indicar o custo como relevante, o assistente sugerirá o Phred como melhor opção. No exemplo, a escolha foi pelo Phred.

No próximo passo, o pesquisador precisa selecionar qual a classe de processos mais adequada para montar os *reads*. CAP3 e Phrap são as classes projetadas para desempenhar esta tarefa. O pesquisador pode selecionar uma destas classes diretamente no *Process View* ou pode indicar nos sub-painéis a esquerda a classe de dados de entrada, de saída e do projeto em que trabalha. Se ele indicar *Read Set* como a classe de dados de entrada, o assistente apresentará um grande número de classes de processos que aceitam esta classe de dados como entrada, de forma direta ou indiretamente. Por exemplo, o sistema incluirá a classe *Sequence Assembly* na lista, já que suas instâncias aceitam conjunto de *reads* como entrada, e também a classe *Sequence Alignment*, que aceita qualquer conjunto de sequências como entrada, o que inclui os *reads*. Se o pesquisador indicar como classe de saída *Contig Set*, o assistente restringirá as possibilidades e apresentará somente uma árvore com as classes de *Sequence Assembly*, neste caso, Phrap e CAP3. Caso ainda exista dúvida, o pesquisador pode selecionar a classe de projeto em que trabalha. Como este exemplo trata de um projeto genoma completo, o assistente informará que o Phrap é a opção mais indicada.

O objetivo do próximo passo é a predição de genes, que pode ser feita utilizando instâncias de processos das classes Glimmer e ORF Finder. O pesquisador pode indicar *Contig Set* como classe de dados de entrada, *ORF Nucleotide Sequence Set* como classe de dados de saída e *Complete* como classe de projeto. O assistente sugerirá o Glimmer como melhor opção.

A tarefa seguinte trata da comparação das ORFs, geradas pelo passo anterior, com NR, um banco de dados público de sequências de aminoácidos. O

pesquisador pode escolher *ORF Amino acid Sequence Set* como classe de dados de entrada no *Input View*, *NR* como classe de recurso utilizando o *Resource View* e *Alignment Set* como classe de dados de saída no *Output View*. O assistente mostrará as classes de processos *Sequence Alignment/Global Alignment/ssearch* e *Sequence Alignment/Local Alignment/BLAST* e *FAST*. O pesquisador terá que escolher entre o alinhamento global e local. Esta escolha é deixada inteiramente para o pesquisador, i.e., ela não depende de classes de dados de entrada, saída, recursos e projetos e nem de propriedades de qualidade. Neste caso, o pesquisador pode ler a descrição das classes e das instâncias de processo clicando com o botão direito do mouse no nome da classe ou da instância no *Processs View* e pedindo para o assistente apresentar uma janela com tal descrição.

Suponha que o pesquisador escolha alinhamento local. Então, ele terá que decidir entre BLAST e FAST. O sistema pode ajudá-lo através das propriedades de qualidade fidelidade e desempenho que diferenciam estas duas classes, pois indicam que as instâncias da classe de processo BLAST são mais rápidas, mas menos precisas que as instâncias da classe de processo FAST. O pesquisador indica pesos para cada qualidade e o assistente calcula a pontuação para cada classe de processo e indica a mais adequada. Suponha que o pesquisador escolha um peso alto para desempenho e baixo para fidelidade. Neste caso, o assistente indicará o BLAST, que possui cinco instâncias de processo. Como a classe de dados de entrada e a classe de recurso são conjunto de sequências de aminoácidos, o assistente indicará o BLASTP como melhor possibilidade.

O pesquisador pode também comparar as ORFs com o banco de dados PFam. Esta comparação pode ser feita com o HMMPFam, que é um outro algoritmo de comparação de sequências que realiza alinhamento local e é mais preciso e menos eficiente que os algoritmos da família BLAST.

Definição de Instâncias de Dados

Ao pedir para incluir uma instância de processo no *Workflow View*, o pesquisador precisa definir algumas informações sobre os dados de entradas, saídas e recursos e suas conexões ao processo. O assistente apresenta uma janela para que estas definições sejam feitas. Esta janela é organizada em forma de tabela, onde cada linha corresponde a uma classe de dado que está associada ao

processo como entrada, saída ou recurso de acordo com a ontologia. As colunas descrevem a instância de dados da seguinte forma:

- *Role*: célula preenchida automaticamente pelo assistente indicando se a instância de dados trata-se de recurso, entrada ou saída. Por exemplo, pode vir preenchido *Input* indicando um dado de entrada.
- *Class*: célula preenchida automaticamente pelo assistente com o nome da da classe de dados da instância, por exemplo *Sequence Set*.
- *Format*: célula preenchida automaticamente pelo assistente indicando o formato interno em que a instância do dado deve estar definida para ser manipulada pela instância do processo, por exemplo *Fasta Sequence Set*.
- *Type*: célula preenchida automaticamente pelo assistente indicando a forma com que a instância será lida (no caso da coluna *Role* ser *Input* ou *Resource*) ou gerada (no caso da coluna *Role* ser *Output*) pela instância do processo. Há duas possibilidades: *gradative* ou *not-gradative*.
- *Value*:
    - No caso da coluna *Role* ser *Output*, o pesquisador deve escrever um nome intuitivo ou deixar o nome padrão escolhido pelo assistente para a instância de dados de saída.
    - No caso de *Role* ser *Input* ou *Resource*, esta célula é uma indicação se as entradas e recursos são saídas de outros processos ou arquivos externos. Se for saída de outro processo, ele deve selecionar o nome de tal saída. Se for um arquivo externo, o pesquisador precisa indicar a opção *New File* e, em seguida, clicar no botão que fará o assistente apresentar uma janela para que usuário escolha o arquivo. Este nome será apresentado em uma célula adicional na mesma linha em que o pesquisador está definindo as informações.

Voltando ao exemplo, quando o pesquisador pediu para incluir a primeira instância de processo Phred no *Workflow View*, o assistente apresenta a Figura 32 descrita anteriormente e que está esquematizada a seguir.

Figura 32. Protótipo: Configuração de dados.

A primeira linha é representante dos cromatogramas que são a entrada do Phred e a segunda é a indicação dos *reads*, saída do Phred. A última coluna mostra que os cromatogramas estão em um arquivo externo, e portanto o pesquisador deve mostrar sua localização, e mostra também que a saída recebe um nome padrão escolhido pelo assistente, neste caso TEMP_1. Este nome pode ser modificado pelo pesquisador para um nome mais intuitivo, como *phred_output*.

Feita a inclusão do Phred, o pesquisador passou para a parte da montagem de fragmentos onde selecionou o Phrap. Nesta etapa é interessante ressaltar que na linha da tabela que trata da instância de dados de entrada do Phrap, o pesquisador deve escolher na coluna *Value* o conjunto de *reads* gerados pelo Phred. Neste momento a coluna *Value* terá uma lista com dois valores *New File* e TEMP_1, sendo que o pesquisador deve selecionar a segunda opção.

As linhas de instâncias de classes de recursos possuem preenchimento análogo às linhas de instâncias de classes de dados de entrada.

Depois da inclusão de uma instância de processo no workflow corrente, o pesquisador pode configurar os seus parâmetros para ajustar a qualidade dos resultados gerados pelo processo. Para tanto, o pesquisador deve clicar com o botão direito no nome da instância do processo e o assistente apresentará uma tabela onde as linhas tratam dos parâmetros de entrada e as colunas de suas propriedades: nome e valor. A Figura 33 ilustra a configuração dos parâmetros de uma instância de processo Phred. A princípio o valor estará configurado como o padrão, mas o pesquisador poderá alterá-lo, e se desistir, retornar para o padrão.

Além disso, o pesquisador também pode redefinir as entradas, saídas e recursos de uma instância de processo presente no workflow corrente, clicando com o botão direito no nome da instância do processo.

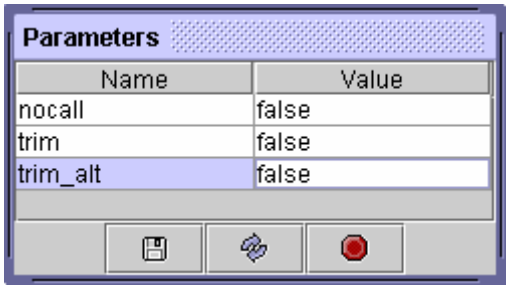

Figura 33. Protótipo: Configuração de parâmetros.

Definição de Instâncias de Processos Filtros

O assistente também apresenta os filtros dos processos de Bioinformática como uma associação pai-filho, ou seja, as instâncias de processos filtros são filhas das instâncias dos processos de Bioinformática no *Process View*. A Figura 31 ilustra que o BLASTP possui dois filtros: MSPcrunch [Sonnhammer, 1994] e BATS [Cavalcanti et al., 2005].

Para incluir um filtro, o pesquisador precisa selecionar a instância de processo filtro no *Process View*, selecionar a instância do processo de Bioinformática no *Workflow View* e clicar no botão do *Process View* para realizar a inclusão. O assistente não permitirá a inclusão de uma instância de processo filtro como filha de uma instância de processo de Bioinformática no *Workflow View* se não houver a associação pai-filho entre estas duas instâncias no *Process View*. Neste exemplo, o pesquisador pode incluir a instância de processo MSPcrunch para filtrar os resultados gerados pelas instâncias de processos da classe BLAST.

Definição de Instâncias de Processos de Controle

No protótipo implementado, o assistente apresenta "!" como uma instância de processo de controle. Este controle interrompe a execução do workflow para que o pesquisador faça uma análise dos resultados intermediários e retorne a execução caso queira.

7.3 Gerente de Otimização

Ao definir um workflow, o pesquisador deve pedir ao assistente para validá-lo. A validação é responsabilidade do gerente de otimização, que verifica as

entradas e recursos de cada uma das instâncias de processo. No protótipo, o gerente de otimização foi implementado instanciando a classe *PersonalOptimizationManager*, ilustrada na Figura 14.

Suponha a análise do processo p_i. Se a entrada ou recurso foram definidas como um arquivo externo, é feita a verificação se o pesquisador definiu o nome e o caminho do arquivo.

Se foram definidas como saída de um outro processo p_j, verifica-se se este processo será executado antes de p_i. Isto é importante porque quando o pesquisador define as instâncias de entrada ou recurso, o assistente apresenta na coluna *Value* na tabela de entradas, saídas e recursos, uma lista completa com todas as saídas geradas por todas as outras instâncias de processos definidas no workflow, independente da ordem. Desta forma, o pesquisador pode, por engano, escolher a instância de entrada de um processo p_i como uma saída de uma instância de processo p_j que será executado após p_i. Além disso, o pesquisador pode ter decidido remover p_j do workflow após a definição de sua instância de saída como instância de entrada de p_i, e tenha se esquecido de redefinir a instância de entrada de p_i.

Se as entradas e recursos definidos como saída de outras instâncias estiverem coerentes, o gerente de otimização passa para a validação dos seus formatos. Caso o formato da instância de saída seja idêntico ao da instância de entrada que foi associada, a definição do pesquisador está correta e não há nada a ser feito. Em caso contrário, verifica-se se existe uma instância de processo de transformação de formato.

No exemplo que está sendo tratado, suponha a definição da instância de processo BLASTP, que realiza a comparação de ORFs com o banco de dados NR. Nesta etapa, o pesquisador precisou definir que a instância de dados de entrada do BLASTP é igual a instância de dados de saída do Glimmer. No entanto, a instância de entrada do BLASTP é um conjunto de sequências de aminoácidos no formato FASTA, enquanto a instância de saída do Glimmer é um conjunto de sequências de nucleotídeos em um formato específico do Glimmer. Neste caso, o gerente de otimização deve incluir uma instância de processo transformadora de formato destas duas instâncias. Atualmente já existem diversos programas, como o EMBOSS/Transeq, que fazem a transformação de sequências de nucleotídeos em aminoácidos. Entretanto, este caso é ainda um pouco mais complexo porque é

necessário a transformação do formato sintático gerado/ utilizado por estas instâncias de processo.

Se o resultado da validação for positivo, o assistente inclui os contêineres para a comunicação dos dados entre as instâncias de processos. No protótipo foi adotada a estratégia da inclusão de um contêiner para cada comunicação entre um par de instâncias de processos. Feito isso, o assistente libera o botão para o pesquisador pedir para criar o documento de especificação do workflow corrente.

No protótipo, o gerente de otimização cria um documento de especificação do workflow como um documento XML seguindo o XML Schema do Quadro 7.

No exemplo do workflow do genoma completo, este documento mostrará que as duas últimas comparações realizadas pelo BLASTP e HMMPFam podem ser executadas em paralelo e há o pipelining do Glimmer e do BLASTP, ou seja, as ORFs resultantes do Glimmer podem ser imediatamente passadas para o BLASTP, sem ter que esperar que o Glimmer produza o conjunto completo de saída.

No protótipo, o assistente apresenta ao pesquisador o documento de especificação do workflow como um documento XML e como uma árvore, e permite a gravação de um arquivo em disco para uma posterior execução através do menu principal do assistente.

7.4 Gerente de Execução

O protótipo implementou o módulo de gerente de execução instanciando a classe *PersonalExecutionManager*, para o próprio gerente de execução, e a classe *XMLWorkflowDefinitionHandler,* para permitir a interpretação do documento de especificação do workflow escrito em XML. Estas classes são ilustradas na Figura 15.

A execução no protótipo de implementação analisa o documento de especificação do workflow e dispara *threads* que simulam as instâncias de processos definidas no workflow utilizando estruturas de dados que simulam a comunicação de dados entre estas instâncias.

A simulação é feita através da produção e do consumo de dados entre as diferentes *threads*. Durante a simulação, dependendo do documento de especificação do workflow definido pelo gerente de otimização, o gerente de

execução dispara as *threads* em paralelo, simulando a execução concorrente de processos e o pipelining das instâncias.

O procedimento de criação de uma *thread* para o tratamento de execução de um processo é dependente do ambiente em que o SGWBio está sendo implementado. Entretanto, em todos os casos, o gerente de execução fornece à *thread* as listas dos parâmetros e dos contêineres de entrada e de saída do processo, de acordo com a definição do documento de especificação do workflow. Esta *thread* é responsável por um procedimento adaptador, que sabe interpretar as informações obtidas do gerente de execução e adaptá-las de forma a executar a instância do processo.

No protótipo, o gerente de execução simula os algoritmos definidos no Capítulo 6, considerando que não há limite de espaço em disco. Desta forma, todos os processos que fazem parte do *estágio_ideal* são disparados de forma concorrente.

A forma com que o gerente de execução foi feito permite sua extensão para que seja utilizado em qualquer ambiente, de acordo com o que foi apresentado no Capítulo 6.

7.5 Gerente de Ontologia

O gerente da ontologia implementado está preparado para fazer acesso a ontologia, independente de sua localização, ou seja, a ontologia pode estar definida em um arquivo texto simples em qualquer formato pré-definido ou em um sistema gerenciador de banco de dados. Neste caso, o protótipo possui a ontologia definida em Amzi-Prolog, apresentada no anexo. Esta decisão foi tomada devido a facilidade da execução de regras de inferência, da legibilidade dos conhecimentos que eram necessários ser descritos, e da existência de uma API de comunicação entre Amzi-Prolog e Java, linguagem escolhida para implementar o sistema.

O protótipo implementou o módulo gerente de ontologia instanciando as classes e sub-classes de *AMZIPrologoOntologyManager* ilustradas na Figura 13. Como exemplo, suponha a sub-classe *SProcessAmziOntMng*, que possui os métodos para acessar as informações na ontologia que dizem respeito às classes de processos de Bioinformática. O método *is_bioinformatics_process* obtém a

lista de todos os processos de Bioinformática que estão definidos na ontologia, enquanto o método *is_bioinformatics_process_filter* descobre se um filtro definido pelo pesquisador para uma instância de processo pode realmente ser aplicado à esta instância.

7.6 Comentários Finais

O protótipo foi desenvolvido para o ambiente pessoal, mas serve de modelo para implementações contemplando os outros ambientes de trabalho definidos no Capítulo 6.

As funcionalidades dos módulos assistente, gerente de ontologia e gerente de otimização são, em grande parte, independentes do ambiente.

O módulo gerente de execução está projetado e implementado com os aspectos independentes das arquiteturas, de forma que é possível estendê-lo para implementar todas as funcionalidades definidas no Capítulo 6, e testar todas as estratégias que foram apresentadas.