

“O AMBIENTE 10+C PARA A DEFINIÇÃO E EXECUÇÃO DE
WORKFLOWS *IN SILICO* ATRAVÉS DE SERVIÇOS WEB”

Rafael Targino dos Santos

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS
PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE
FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS
NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM
ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:



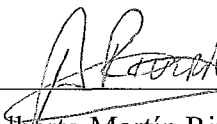
Prof. Marta Lima de Queirós Mattoso, D.Sc.



Prof. Geraldo Bonorino Xexéo, D.Sc.



Prof. Carla Osthoff Ferreira de Barros, D.Sc.



Prof. Alberto Martín Rivera Dávila, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

NOVEMBRO DE 2004

SANTOS, RAFAEL TARGINO DOS

O Ambiente 10+C para definição e execução de workflows *in silico* através de serviços web [Rio de Janeiro] 2004

VIII, 118 p., 29,7 cm (COPPE/UFRJ, Mestre, Engenharia de Sistemas e Computação, 2004)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. BANCO DE DADOS

2. WORKFLOWS

3. BIOINFORMÁTICA

I. COPPE/UFRJ II. Título (série)

“Valorize suas limitações e terá de conservá-las”.

Richard Bach - Ilusões

Em 2001, quando iniciei os estudos para o mestrado e ainda nem tinha idéia do que seria a minha dissertação, a seção de “agradecimentos” sempre me despertou grande atenção e eu me perguntava o que iria escrever quando chegasse a minha vez de agradecer. Hoje, alguns anos depois, após ler várias seções de agradecimentos de outras teses, noto que a essência delas sempre é a mesma, e eu, não tenho a menor pretensão em modificar essa prática. Por isso, aqui vão os meus mais sinceros agradecimentos, que apesar de serem os usuais, são os mais verdadeiros possíveis.

Em primeiro lugar gostaria de agradecer a minha orientadora Marta Mattoso por me iniciar na pesquisa científica, me incentivar nos estudos e sempre buscar o melhor para esta dissertação. Ao grupo de Bioinformática da COPPE/Sistemas, a quem devo enorme gratidão pela ajuda e conselhos no desenvolvimento desta dissertação, especialmente Maria Cláudia Cavalcanti (Yoko), Fabrício Teixeira, Fernanda Baião, Paulo Pires, Luiz Vivacqua, Shaila Rössle e Paulo Bisch.

Aos membros da banca, professores Geraldo Xexéo, Carla Osthoff e Alberto Dávila, pela apreciação deste trabalho, críticas e comentários. Ao professor Jano de Souza, pela condução do grupo de Banco de Dados da COPPE/Sistemas.

Ao CNPq, pela bolsa de mestrado e à Fundação COPPETEC pelas oportunidades nos projetos SEPS da Embratel e SINGRA da Marinha, que constituíram o apoio financeiro essencial para que eu tivesse tempo para dedicar-me aos estudos. Mas especialmente, ao Prof. José Roberto Blaschek, pelos ensinamentos, conselhos, incentivo e oportunidades nesses últimos seis anos. À Patrícia Leal pelo apoio e disposição em sempre resolver os problemas administrativos.

Aos inúmeros amigos e colegas, que ajudaram, direta ou indiretamente, na forma de um conselho sobre este trabalho ou pelos momentos de distrações necessárias, em especial a minha turma de mestrado da COPPE, aos amigos da graduação da UFRJ, ao pessoal do futebol e chopp de segunda-feira, aos amigos da turma do 2º grau do Bennett e aos amigos do projeto SEPS da Embratel e SINGRA da Marinha.

Ao inventor do Café (como bebida), combustível necessário aos meus estudos, ao Maracanã, pelas tardes de domingos emocionantes vendo o meu Fluminense, à Internet, fonte maior de pesquisa, ao Vinho e ao Chopp, essenciais nos momentos de relaxamento e ao Restaurante Burguesão, local preferido para as minhas leituras acompanhado de um bom expresso.

A toda a minha família, em especial meu pai Solimar e minha mãe Eunice pelo imenso amor e constante incentivo na minha formação como pessoa e como profissional. À minha irmã Lisiane, pelo grande exemplo de garra e perseverança de sua vida, e que apesar de todos os quilômetros de distância, sempre procurou estar presente. À minha namorada Bárbara, pelo amor, apoio, incentivo e por dividir os diversos momentos, altos e baixos, que ocorreram durante o desenvolvimento deste trabalho.

A Deus, por tudo.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

O AMBIENTE 10+C PARA DEFINIÇÃO E EXECUÇÃO DE WORKFLOWS *IN SILICO* ATRAVÉS DE SERVIÇOS WEB

Rafael Targino dos Santos

Novembro/2004

Orientadora: Marta Lima de Queirós Mattoso

Programa: Engenharia de Sistemas e Computação

Cientistas de todo o mundo estão cada vez mais efetuando experimentos científicos totalmente executados e analisados através de computadores. Grande parte dos chamados experimentos *in silico* correspondem à composição de vários programas em seqüência, onde a saída de um deles é utilizada como entrada de dados do próximo, com a utilização de um grande número de bancos de dados. Normalmente, esses diversos programas são executados com controle manual pelos cientistas ou através do uso de linguagens de *scripts*, como Perl. Este tipo de abordagem ajuda na automatização da cadeia de execução, mas possui grande deficiência em questões como flexibilidade, interoperabilidade, clareza, registros de uso, manutenção e evolução.

Workflows científicos representam uma alternativa atraente para a descrição deste tipo de experimento, além de fornecer o apoio necessário ao ciclo de “execução e análise” inerente ao processo de busca de conhecimento. Aliados à tecnologia de serviços Web, pode-se criar um ambiente com independência e interoperabilidade entre as diversas aplicações científicas e os diversos bancos de dados. Apesar do uso bem sucedido dessas tecnologias na área comercial, seu uso em bioinformática ainda é incipiente. Neste contexto, o objetivo desta tese é a elaboração, prototipação e avaliação de um ambiente integrado, chamado de Ambiente 10+C, para a definição e execução de experimentos *in silico* através de workflows científicos compostos de diversas aplicações disponíveis como serviços Web. Um experimento real de bioinformática, chamado MHOLline, foi implementado neste ambiente para validar e confirmar a proposta desta tese e analisar a viabilidade de sua implementação em um sistema de produção.

Abstract of Thesis presented to COPPE/UF RJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

THE 10+C ENVIRONMENT TO DEFINE AND EXECUTE
IN SILICO WORKFLOWS USING WEB SERVICES

Rafael Targino dos Santos

November/2004

Advisor: Marta Lima de Queirós Mattoso

Department: Computer Science and Systems Engineering

Scientists around the world are developing scientific experiments that are executed and analyzed using computers. A great amount of the so called *in silico* experiments correspond to a composition of sequences of several programs where the output of one program is used as data input to the next program accessing several databases. Usually, these programs are executed without automation by the scientists or through the use of script languages such as Perl. The script approach helps the automation of the execution process, however it has many deficiencies in terms of flexibility, interoperability, clarity, execution registers, maintenance and evolution.

Scientific workflows represent an attractive alternative to describe this kind of experiments, as well as to give the necessary support to the “Execution and Analysis” cycle, relevant to the process of knowledge discovery. Workflows can create an independent and interoperable environment between the scientific applications and databases, when combined with the Web Services technology. In spite of the successful use of these technologies in the business scenario, its use in bioinformatics is still incipient. In this context, the objective of this thesis is the elaboration, prototyping and evaluation of the integrated environment called the 10+C Environment. 10+C aims at the definition and execution of *in silico* experiments through scientific workflows made of several Web services applications. A real bioinformatics experiment named MHOLline was implemented in this environment in order to validate and confirm the goals of this thesis and to analyze the viability of its implementation in a production system.

CAPÍTULO 1	1
1.1 OBJETIVOS	5
1.2 ORGANIZAÇÃO DOS CAPÍTULOS	11
CAPÍTULO 2	13
2.1 FUNDAMENTOS DA BIOLOGIA MOLECULAR	14
2.2 BIOINFORMÁTICA	17
BANCO DE DADOS DE BIOLOGIA MOLECULAR	20
COMPARAÇÃO E ANÁLISE DE SEQUÊNCIAS	22
PREDIÇÃO DE ESTRUTURA DE PROTEÍNAS	24
2.3 PRINCIPAIS DESAFIOS	25
GRANDE HETEROGENEIDADE DE BANCO DE DADOS E APLICAÇÕES	26
NECESSIDADE DE INTEGRAÇÃO ENTRE DIVERSOS BANCOS DE DADOS	28
DEFINIÇÃO E EXECUÇÃO DE EXPERIMENTOS ATRAVÉS DA COMPOSIÇÃO DE APLICAÇÕES CIENTÍFICAS	30
CAPÍTULO 3	35
3.1 SERVIÇOS WEB	35
3.2 WORKFLOWS	40
LINGUAGENS DE DEFINIÇÃO DE WORKFLOWS	43
WORKFLOWS CIENTÍFICOS	44
3.3 10+C: AS 10 PRINCIPAIS CARACTERÍSTICAS PARA AMBIENTES DE EXPERIMENTOS <i>IN SILICO</i>	46
3.4 TRABALHOS RELACIONADOS	50
LABBASE/LABFLOW	51
CHIMERA	52
MYGRID	54
ARSENAL	56
GRNA	57
BIOMOBY	59
SRMW	59
DISCUSSÃO SOBRE OS TRABALHOS RELACIONADOS	62
CAPÍTULO 4	64
4.1 ARQUITETURA	65
4.2 PARTICIPANTES DA ARQUITETURA	68
4.3 CICLO DE VIDA	68
O AMBIENTE 10+C E A SRMW	71
4.4 O WORKFLOW MHOLLINE	72
4.5 IMPLEMENTAÇÃO DO MHOLLINE NO AMBIENTE 10+C	74
APLICAÇÕES CLIENTE	76
SERVIÇOS WEB DE APLICAÇÕES CIENTÍFICAS	77
SERVIÇOS WEB DE PERSISTÊNCIA DE DADOS	79
MÓDULO DE DEFINIÇÃO DE WORKFLOWS	80

MÓDULO DE EXECUÇÃO DE WORKFLOWS	86
CAPÍTULO 5	89
5.1 ATENDIMENTO ÀS CARACTERÍSTICAS 10+C	90
5.2 AVALIAÇÃO QUALITATIVA	92
O CAOS DOS <i>SCRIPTS</i>	92
COMPARAÇÃO DO EXPERIMENTO MHOLLINE	94
5.3 AVALIAÇÃO QUANTITATIVA	98
CAPÍTULO 6	101
CAPÍTULO 7	108

“Um ovo, um embrião, um adulto - é o normal. Mas um ovo bokanovskizado tem a propriedade de germinar, proliferar, dividir-se: de oito a noventa e seis gemes, e cada um destes se tornará um embrião perfeitamente formado, e cada embrião, um adulto completo. Assim se consegue fazer crescerem noventa e seis seres humanos em lugar de um só, como no passado. Progresso”.

Aldus Huxley - Admirável Mundo Novo

Há 50 anos atrás, em 25 de abril de 1953, quando Francis Crick e James Watson anunciaram a descoberta da estrutura básica do **DNA** [169], a molécula de dupla hélice que guarda todas as informações genéticas dos organismos [63], e entraram para a história com uma das maiores descobertas científicas da humanidade, valhendo-lhes o Prêmio Nobel de Medicina em 1962, provavelmente não imaginaram todas as possibilidades que sua pesquisa representava [156].

Atualmente, as pesquisas de biologia molecular representam uma área em enorme evidência no mundo inteiro e os seus resultados podem ser facilmente verificados na sociedade em geral. Testes de paternidade em programas de televisão, a utilização de alimentos transgênicos nos noticiários e a velha polêmica a respeito dos clones humanos têm se tornado assuntos rotineiros na vida das pessoas.

Grande parte deste avanço da pesquisa de biologia molecular pode ser creditado à intensa utilização de recursos computacionais no dia a dia dos cientistas e centros de pesquisas, no que se convencionou chamar de **bioinformática**. A automatização dos procedimentos de coleta, representação, armazenamento e análise dos dados originados do uso de computadores e aplicações de software, além do surgimento e consolidação da Internet, foi o que permitiu estabelecer o patamar atual da pesquisa genética nesta última metade de século.

Embora toda essa automatização seja bastante benéfica permitindo a otimização na execução dos diversos procedimentos e aumentando a qualidade dos resultados obtidos, ela também criou diversos novos desafios que devem ser tratados pela comunidade científica. O crescimento exponencial do uso de recursos computacionais gerou um ambiente caracterizado pela grande heterogeneidade e falta de integração entre as diversas aplicações e bancos de dados existentes nos diferentes centros de pesquisa.

Atualmente, existem mais de 280 bancos de dados de biologia molecular [23] para os mais diferentes organismos. Até mesmo as aplicações, possuem diferentes implementações de um mesmo algoritmo em diferentes centros de pesquisa [31][168]. A difusão de aplicações com propósitos semelhantes mas específicas para determinados bancos de dados dificultam o trabalho dos biólogos. Além disso, muitos desses programas e fontes de dados ou são incompatíveis entre si, no que diz respeito a aspectos tecnológicos como sistemas operacionais e protocolos de rede, ou são complementares quanto à sua utilização em conjunto, ou ambos.

Diversas iniciativas surgiram para tratar o problema da integração de bancos de dados de biologia molecular, desde abordagens simplistas que utilizam ligações entre registros de diferentes fontes de dados (SRS [62], LinkDB [67]) até abordagens baseadas em mediadores como K2/Kleisli [55] e TAMBIS [123] ou ainda abordagens baseadas em repositórios centrais materializados a partir de diversas fontes como GUS [55] e GIMS [50]. No que se refere às aplicações, algumas iniciativas possuem a proposta de disponibilizar diversos algoritmos comuns em pacotes reutilizáveis de código aberto como bioPerl [146] e bioJava [145], enquanto outras foram desenvolvidas mais voltadas a padronização de diferentes recursos de bioinformática como o DAS [54] e o I3C [89].

Embora todos esses esforços sejam extremamente válidos no acesso e manipulação de dados integrados, permanece ainda uma grande lacuna quando se deseja um nível maior de integração, não apenas entre as diversas fontes de dados existentes, mas também entre a execução de programas e ferramentas que executam **transformações** sobre esses dados. A execução de diversas transformações sobre os dados, é um processo cada vez mais utilizado na pesquisa genômica, no que se convencionou chamar de **experimentos *in silico***. A principal característica desse tipo de

experimento é a gradativa complementação das etapas *in vitro* produzidas nos laboratórios de pesquisa, por experimentos biológicos totalmente executados e analisados em computadores. Assim, ao invés de tentar estabelecer a estrutura da proteína através de experimentos em laboratórios, um biólogo poderia, usar um experimento como o MHOLline [130], para a partir da seqüência de aminoácidos da proteína, 1) utilizar o programa BLAST para encontrar seqüências semelhantes; 2) procurar em um banco de dados como o PDB se estas seqüências possuem estrutura conhecida; 3) construir modelos a partir das seqüências que possuem estruturas conhecidas e; 4) construir a estrutura tridimensional da proteína pesquisada a partir dos modelos gerados, utilizando um programa como o Modeller. Por fim, 5) o resultado final poderia ainda ser validado utilizando algum programa como o Procheck que verifica se a estrutura encontrada é estável, conforme (Figura 1).

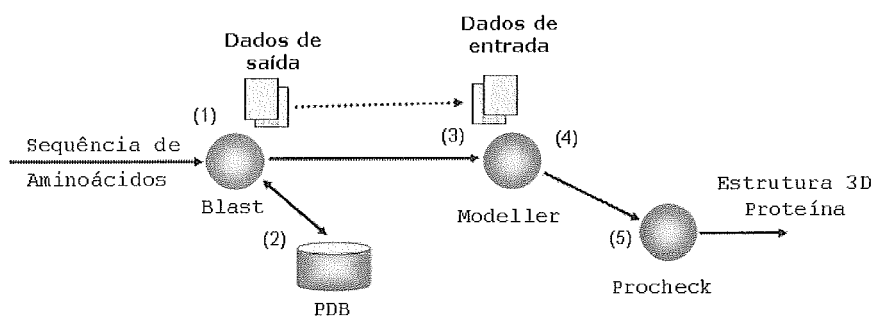


Figura 1 - Etapas de um experimento *in silico* que utiliza os programas BLAST, Modeller e Procheck além do banco de dados PDB para o processamento de uma seqüência de aminoácidos de entrada [130].

Na maioria das vezes, estes experimentos são feitos através da execução de diversos programas em seqüência, normalmente disponibilizados na Internet como executáveis ou páginas Web, onde a saída de um deles é utilizada como a entrada para outro, de forma que os vários programas sejam compostos como se fosse uma única aplicação. Para isto acontecer de forma automática, a **interoperabilidade** entre os vários programas e bases de dados existentes é um requisito fundamental e para atender este objetivo também será necessário algum mecanismo que padronize e permita a comunicação entre esses diversos recursos. Na computação, o arcabouço que permite a **composição de programas** em uma seqüência de execução com o objetivo de gerar um resultado final é chamado de **workflow**. Um workflow provê a abstração necessária para descrever uma série de atividades e processos estruturados com o objetivo de prover um ambiente robusto de resolução de problemas e assim, habilitar o uso efetivo dos recursos computacionais. Dessa forma, workflows podem ser utilizados tanto para

modelar a definição de experimentos *in silico* (seqüência de etapas de execução que podem possuir testes, desvios, tratamento de erros, etc.), quanto para a execução e monitoramento do mesmo (a execução de uma instância do workflow previamente definido, passando os dados e parâmetros de entrada resultando em uma saída final). O termo **workflow científico** é utilizado, em oposição aos workflows comerciais usados nos processos de negócios das empresas, para descrever a classe de workflows mais voltada na resolução de problemas científicos, que normalmente possuem maior complexidade, maior necessidade de processamento de alta capacidade e maior número de execuções a cada simulação do experimento.

Atualmente, em grande parte dos casos, esta composição de programas é feita manualmente pelos biólogos, que executam um programa após o outro. Os dados gerados são analisados individualmente, e conforme o resultado da análise, o biólogo escolhe o próximo programa do fluxo de execução, copia e converte os dados anteriores para a nova execução (pois normalmente os formatos utilizados pelos programas não são compatíveis) e assim dá continuidade ao seu experimento. Muitas vezes, os biólogos utilizam alguma linguagem de programação de *script* como Perl [124] para auxiliar na tarefa de composições de programas, cópias e conversão de dados, mas esta abordagem além de ser trabalhosa, possui grande dependência do ambiente (sistema operacional, localização dos programas e dados, direitos de execução sobre arquivos e diretórios), grande dificuldade de construção (criação e manutenção da definição do workflow, descoberta do programa correto a ser utilizado), além de grande dificuldade de utilização (definição dos parâmetros de uso, utilização simultânea por muitos usuários, grande quantidade de execuções, portabilidade para outros ambientes). A este ambiente, adicionam-se ainda outras deficiências como a ausência de registro das execuções de programas (o que leva a perda da experiência sobre os experimentos) e a grande limitação no uso das aplicações científicas (já que elas precisam estar instaladas localmente ou acessíveis na rede interna do laboratório).

Toda essa rigidez própria dos ambientes de programação de linguagens scripts tem levado diversos centros de pesquisa em bioinformática a utilizar a tecnologia de **serviços Web** (*Web Services*) [160] em alguns de seus projetos, mas ainda há uma grande carência de experimentos reais de comparação entre esses dois ambientes. A tecnologia de serviços Web provê os mecanismos necessários para habilitar a interoperabilidade entre recursos heterogêneos na Web, permitindo a padronização na

comunicação entre os diversos programas e dados de bioinformática existentes. Uma das principais características desta tecnologia é o uso de protocolos e linguagens baseadas em XML para a especificação de interfaces, como a WSDL (*Web Services Description Language*) [161] para a descrição de serviços e o SOAP (*Simple Object Access Protocol*) [162] como protocolo de comunicação. Uma vez que o recurso esteja publicado como um serviço Web, ele poderá ser acessado por qualquer tipo de programa cliente na Internet, através de uma interface bem definida, tornando assim, cada recurso em um módulo de software totalmente reutilizável.

1.1 Objetivos

Neste contexto, o objetivo principal desta dissertação é a elaboração, prototipação e avaliação de um ambiente integrado para a definição e execução de experimentos *in silico* através de workflows científicos compostos de diversos programas e fontes de dados disponíveis como serviços Web. A este ambiente, batizamos com o nome de Ambiente 10+C, pois o mesmo foi orientado para atender dez características significantes a este tipo de problema.

Dessa forma, através de serviços Web, cabe ao biólogo especificar e montar a definição do workflow que atenda a sua pesquisa, assim como a execução de várias instâncias deste workflow, definindo parâmetros e dados de entrada ao longo do tempo. Nosso ambiente, portanto, pretende ser genérico e flexível suficiente, para permitir a personalização, adicionando e integrando novos recursos de bioinformática conforme as necessidades e permitindo assim, a criação de autênticos workflows científicos de bioinformática.

Assim, a definição do workflow pode ser feita através de um conjunto de páginas Web que fornece os recursos necessários para escolher o serviço Web desejado, montar o fluxo de execução do experimento e mapear as saídas em correspondentes entradas de dados dos próximos programas. A utilização deste tipo de recurso permite uma maior facilidade na construção de um workflow, já que dessa forma não é necessária a alteração de qualquer linha de código, o que normalmente acontece quando a definição do workflow está embutida junto com a linguagem de programação.

Uma vez definido, o workflow pode ser instanciado diversas vezes para cada experimento, modificando-se apenas os dados e parâmetros de entrada. Assim, durante a execução de um experimento de cultura de bactérias, por exemplo, bastaria apenas executá-lo seguidas vezes mudando apenas os valores para o pH do experimento. Para cada execução, tanto os parâmetros e os dados de entrada, como os dados intermediários, seriam armazenados em um banco de dados para permitir uma melhor análise do experimento. O armazenamento desses dados traz pelo menos duas vantagens importantes: permite a identificação da procedência e das transformações sobre os dados (*data provenance*) [34][35] e permite a possibilidade de re-execuções totais ou parciais do experimento.

A identificação da procedência e das transformações efetuadas sobre os dados é de vital importância para a confiabilidade que este dado terá. Dessa forma, a informação sobre procedência de um dado poderia ser armazenada como “anotações”, uma forma amplamente utilizada na biologia e que poderia auxiliar na busca por novas descobertas. O armazenamento dos dados também permitirá a obtenção de um conhecimento sobre os experimentos que de outra forma eram pouco cogitados e tinham uma grande dificuldade de serem obtidos. Perguntas para descobrir os resultados obtidos há seis meses atrás e como rastrear como dados e transformações foram derivadas podem ser facilmente respondidas. Na verdade, a procedência de dados é uma área muito mais ampla e complexa, e esses são apenas os primeiros passos nesta direção. Além disso, esses dados armazenados também podem ser utilizados para a extração de conhecimento científico adicional, a partir de técnicas de mineração de dados.

A re-execução de experimentos é uma característica muito importante no que se refere à economia de tempo. Sabendo-se que os muitos experimentos biológicos normalmente costumam consumir muito tempo de processamento, a possibilidade de re-executar um experimento a partir de um ponto qualquer, mantendo todos os resultados até este ponto, mas permitindo a modificação dos parâmetros ou dados intermediários a partir dali, representa um enorme aumento na produtividade do trabalho produzido.

Além disso, como todos os dados estão armazenados eles poderiam ser compartilhados entre diversos biólogos, auxiliando o trabalho em equipe, conforme as necessidades de cada centro de pesquisa. Assim, um experimento poderia não ser

apenas executado e analisado por um biólogo, mas sim por toda uma equipe que poderia estar remotamente distribuída.

Para efetuar a construção deste ambiente de definição e execução de workflow, estabeleceu-se necessário utilizar algum experimento real de bioinformática como forma de aplicar e validar a proposta desta dissertação. Para isso, foi escolhido o projeto MHOLline [130] do Instituto de Biofísica Carlos Chagas Filho da UFRJ (IBCCF/UFRJ), como o experimento que serviria como estudo de caso para a aplicação de workflows científicos de serviços Web. O MHOLline é um exemplo de um experimento de bioinformática para a predição de estruturas de proteínas e a conseqüente definição da função dos genes de um organismo. Este tipo de experimento não é fácil de ser construído devido à complexidade dos programas envolvidos e os dados que são trocados entre eles. Todos os programas envolvidos estão disponibilizados na Internet sem nenhum custo comercial e podem ser executados localmente, não possuindo nenhuma interface de serviços Web já pronta. Os programas são executados um após o outro em seqüência e existe um *script* Perl que é responsável pela invocação de cada um dos programas, por pequenas transformações feitas nos dados e ainda pela adição de uma nova funcionalidade (um novo programa) que estende um dos principais programas utilizados.

Dessa forma, foi necessário em um primeiro momento, disponibilizar as aplicações na forma de serviços Web, para só então, efetuar a composição dessas aplicações em um workflow científico. Neste contexto, o desenvolvimento desta dissertação pode ser dividido basicamente em cinco etapas:

- (i). Disponibilização de aplicações públicas de bioinformática através de serviços Web;
- (ii). Definição de uma forma para descrever a composição desses serviços Web como workflows científicos;
- (iii). Execução do workflow definido previamente;
- (iv). Construção de um ambiente web para melhor interação com (ii) e (iii), e
- (v). Comparação dos resultados da execução do workflow de serviços Web com o workflow normal escrito em Perl.

Como dito anteriormente, a maioria dos programas de bioinformática estão disponíveis na Internet ou como programas executáveis ou na forma de formulários

HTML, mas nenhum deles na forma de serviços Web. A primeira etapa desta dissertação constituiu-se de construir os serviços Web para o conjunto de aplicações utilizadas no experimento MHOLline. Programas como BLAST [8], BATS [130], Modeller [132] e Procheck [97] foram disponibilizados nesta nova tecnologia através da construção da interface WSDL/SOAP.

A composição dos diversos serviços Web em workflows científicos foi efetuada utilizando uma linguagem de definição de processos de workflows. Das diversas propostas existentes atualmente, foi escolhida a linguagem de definição BPEL4WS – *Business Process Execution Language for Web Services* [52], ou abreviadamente BPEL, patrocinada por grandes fornecedores como IBM, Microsoft e BEA. O BPEL é uma linguagem baseada em XML para a coordenação de processos sobre a Internet, e constitui-se atualmente, em um forte candidato a ser transformado num padrão de fato para composição de serviços Web. Com a utilização do BPEL, a seqüência de execução de programas pode ser descrita em um documento XML que é totalmente independente da execução do workflow propriamente dito. O mais interessante desta proposta, é que o workflow também é disponibilizado como um serviço Web, desfrutando de todos os benefícios que esta tecnologia oferece e ainda, podendo até mesmo ser utilizado como parte de outros experimentos.

A terceira etapa desta dissertação foi executar o workflow definido anteriormente. Para isso, foi utilizado um mecanismo de execução (*engine*) open-source executora do BPEL disponibilizada pela IBM [32]. Este mecanismo processa os arquivos XML e é responsável por invocar cada um dos serviços Web descritos na definição do workflow e transferir os dados entre eles.

A quarta etapa foi a construção de um portal Web para dar suporte às duas etapas anteriores: a definição e a execução de workflows. Através deste portal, um workflow pode ser elaborado e executado com poucos conhecimentos de tecnologias como de serviços Web ou workflows, ou seja, ideal para o uso de biólogos e outros cientistas que não são da área de computação. Esse portal foi construído através de *servlets* Java [91] e o servidor de banco de dados MySQL [111].

Na última etapa, foi executada a comparação entre os dois ambientes de bioinformática. O tradicional, onde as execuções dos programas eram locais e existia

um *script* Perl responsável por isso e o ambiente proposto por esta dissertação, que utiliza workflows científicos e serviços Web para permitir a definição e execução pela Internet. Nesta etapa, foi verificado que apesar do tempo de execução de um workflow de serviços Web ser maior que o da abordagem tradicional, esta diferença pode ser ignorada, pois o tempo de execução das aplicações científicas estão em média duas ordens de grandeza acima do tempo de execução dos workflows.

Diversas abordagens foram propostas na literatura com objetivos próximos ao desta dissertação. Uma das primeiras iniciativas ainda na década de 90 foi o sistema LabBase/LabFlow [73][74] que já previa um ambiente para a definição e execução de workflows, mas em uma ambiente fechado e controlado. As descrições dos programas e dos workflows eram feitas utilizando elementos definidos exclusivamente para este sistema, dificultando aspectos de distribuição e interoperabilidade. O sistema Chimera [66] utiliza a infra-estrutura aberta de Grids Computacionais [65], mas também utiliza conceitos próprios para a descrição de programas e workflows. O sistema provê um mecanismo de execução de workflows, mas falha no registro de execuções e na possibilidade de re-execuções parciais. O projeto myGrid [179] também utiliza a infra-estrutura de Grids com serviços Web e é um dos trabalhos mais completos analisados nesta dissertação. Ele possui um grande suporte na descrição de programas e experimentos através da utilização de ontologias [77][78]. Também possui suporte a descrição dos dados relacionados às execuções dos programas, porém, não foram encontrados registros de utilização de seus serviços Web em experimentos reais de bioinformática.

Outros trabalhos analisados nesta dissertação foram os projetos ARSENAL [96], gRNA [98] e BioMoby [175]. O projeto ARSENAL utiliza serviços Web e workflows para o processamento de análises complexas no campo da astronomia. Ele possui suporte básico para a definição e execução de workflows, mas não oferece funcionalidades para a definição abstrata, para re-execuções e para o registro de programas e experimentos. O gRNA é um sistema completo para a definição e execução de workflows através de uma API específica e um programa chamado HyperThesis [30]. Através deste programa pode-se visualmente efetuar a definição de um workflow e também executá-lo. Porém, os programas que compõem o workflow devem fazer parte desta arquitetura, não aproveitando os diversos programas já existentes. Por último, o projeto BioMoby é todo voltado para programas disponíveis como serviços Web mas

não possui nenhum tipo de mecanismo para a definição e execução de workflows científicos. O BioMoby tem o seu objetivo principal limitado em prover uma maior semântica na descrição de serviços de bioinformática através da utilização de ontologias.

Apesar da gerência de workflows científicos ser importante para a execução de experimentos *in silico*, ela não é o bastante. Para habilitar todo o potencial que um laboratório virtual de experimentos pode ter, ainda se faz necessário uma descrição em um nível de abstração ainda mais alto de programas, dados e workflows de forma a permitir a criação de modelos científicos. Esta descrição mais precisa dos recursos científicos permite um melhor intercâmbio, reuso e disseminação do conhecimento entre equipes de cientistas distribuídas geograficamente. A arquitetura SRMW [43] fortemente apoiada no uso de metamodelos [42] originada da tese de doutorado de Cavalcanti [40] trata o problema de gerenciamento de recursos científicos. Esta arquitetura é inovadora no uso de metamodelos para a descrição de recursos científicos, adicionando mais semântica a eles. Porém, não fez parte do escopo do trabalho de Cavalcanti o suporte à definição e à execução dos workflows descritos. Na verdade, essa ausência ao nível operacional é que motivou o início desta dissertação, de forma a prover um ambiente que combine o suporte de metadados ao gerenciamento de workflows científicos [44].

Esta dissertação também tem estreita ligação com outro trabalho que está sendo desenvolvido na COPPE/Sistemas. O trabalho de dissertação de Mestrado de Teixeira [142] trata do problema de armazenamento e publicação de resultados de workflows através do uso de tecnologia de serviços Web [143] e também poderia ser utilizado no Ambiente 10+C para o registro de dados das execuções de programas e workflows científicos.

Finalmente, mesmo existindo diversas iniciativas de uso de serviços Web e workflows em bioinformática, não foram encontrados trabalhos que analisassem experimentalmente a utilização dessas tecnologias em um experimento científico real, e que comparasse os resultados obtidos com a abordagem tradicional dos biólogos. Sendo assim, essa dissertação contribui através do desenvolvimento do Ambiente 10+C, para mostrar a efetividade desta solução, na definição e execução de um experimento real de bioinformática.

1.2 Organização dos Capítulos

O restante dessa dissertação está organizado da seguinte forma:

O capítulo 2 apresenta uma introdução geral sobre biologia molecular computacional. Os conceitos básicos de biologia molecular e as principais aplicações existentes são apresentados nas primeiras seções. Na última seção deste capítulo, os principais desafios da área de bioinformática são discutidos, destacando-se a grande heterogeneidade de bases de dados e aplicações, a necessidade de integração entre os diversos recursos de bioinformática e a composição de várias aplicações na forma de experimentos *in silico*.

O capítulo 3 apresenta uma introdução sobre as tecnologias de workflows e serviços Web utilizados nesta dissertação. Os conceitos principais, os objetivos e suas principais características são mostrados nas duas primeiras seções do capítulo. Na última seção é feita uma avaliação dos diversos trabalhos relacionados a esta dissertação, destacando a SRMW que motivou o início deste trabalho. Além disso, foi definido um conjunto de dez características relevantes para soluções de workflows científicos de bioinformática, chamadas de características de 10+C.

O capítulo 4 apresenta o Ambiente 10+C desenvolvido nesta dissertação. As primeiras três seções mostram a arquitetura, os participantes e o ciclo de vida de um experimento no Ambiente 10+C. A seção 4.4 apresenta o workflow MHOLline, um experimento real de bioinformática desenvolvido no IBCCF/UFRJ para a predição de estruturas tridimensionais de proteínas. A seção 4.5 mostra como foi feita a implementação do Ambiente 10+C utilizando serviços Web, a linguagem de definição de workflows BPEL e um portal Web escrito em Java e MySQL.

O capítulo 5 desta dissertação é dedicado à avaliação do Ambiente 10+C. Na primeira seção é discutido como nosso ambiente atendeu a cada uma das características 10+C. A seção 5.2 apresenta uma avaliação qualitativa entre a nossa abordagem e a abordagem tradicional de linguagem de script. Na seção 5.3 efetuamos uma comparação entre as duas abordagens através da execução do experimento MHOLline nos dois ambientes.

Finalmente, o último capítulo apresenta a conclusão desta dissertação, resumindo seu conteúdo, discutindo as principais contribuições e incluindo algumas perspectivas de trabalhos futuros.

Biologia Molecular Computacional

“Homens e mulheres padronizados, em grupos uniformes. Todo o pessoal de uma pequena usina constituído pelos produtos de um único ovo bakanovskizado. Noventa e seis gêmeos idênticos fazendo funcionar noventa e seis máquinas idênticas. (...) O princípio da produção em série aplicado enfim a biologia”.

Aldus Huxley - Admirável Mundo Novo

A ciência mudou depois da descoberta da estrutura básica do DNA, a molécula considerada como o “Livro da Vida”. Os avanços na área da saúde proporcionados pela descoberta do DNA são muitos e cada vez mais presentes em nossa sociedade. Neste capítulo é apresentada uma introdução inicial sobre a base teórica para o entendimento do código genético e do DNA, assim como os genes e as proteínas que são geradas a partir deles. Também são apresentadas as principais aplicações existentes relacionadas à biologia molecular computacional, destacando a busca de seqüências em bases de dados genômicas, a comparação e análise de seqüências, e a predição de estrutura de proteínas, que é o assunto do estudo de caso deste trabalho. Por fim, são discutidos os principais desafios atuais da área de bioinformática que envolvem a grande heterogeneidade de bases de dados e aplicações, a necessidade de integração entre os diversos recursos de bioinformática e principalmente, a composição das várias aplicações existentes na forma de experimentos *in silico*, que é o foco desta dissertação.

2.1 Fundamentos da Biologia Molecular

Todos os seres vivos, com exceção dos vírus, são constituídos por células [9]. Apesar de diferentes em inúmeros aspectos, as células dos diversos organismos, sejam de animais, vegetais, bactérias, algas, fungos ou protozoários, são compostos basicamente pelas mesmas substâncias, e seus processos e reações vitais são bem semelhantes. No interior dos núcleos das células estão presentes finíssimos filamentos enovelados chamados de **cromossomos**. O número de cromossomos é característico em cada espécie, como por exemplo, o homem apresenta 46 cromossomos, as moscas 10 e a cebola 16. Na maioria das células, os cromossomos estão presentes em pares que possuem as mesmas características e funcionalidades, sendo chamados de cromossomos homólogos. Assim, o homem possui 23 pares de cromossomos homólogos ou genericamente $2n$.

Cada cromossomo é constituído por uma molécula de ácido desoxirribonucléico, o **DNA** ou **ADN**, que é formado pela união de três tipos de substâncias: ácido fosfórico, o açúcar desoxirribose e bases nitrogenadas Figura 2. Os nucleotídeos que formam o DNA diferem entre si apenas quanto à base nitrogenada, que pode ser de quatro tipos diferentes: adenina (A), guanina (G), citosina (C) e timina (T). O DNA é formado por duas cadeias de milhares de nucleotídeos que se mantêm paralelas como se fossem os corrimões de uma escada de corda torcida em espiral, em uma estrutura conhecida como dupla-hélice. Essa estrutura se mantêm unida pela fraca ligação entre duas bases complementares: adenina em uma ponta se junta com timina e citosina com guanina [181].

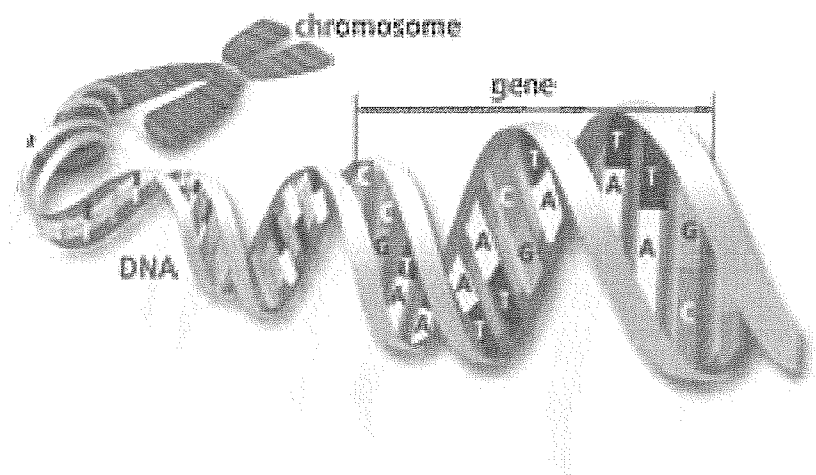


Figura 2 – Estrutura do DNA [157]

As moléculas de DNA são consideradas as mestras da vida pois são elas que possuem as instruções de como produzir todas as partes de um novo ser vivo. Essa capacidade do DNA é exercida através do controle da **síntese de proteínas**, o componente básico de todos os seres vivos Figura 3.

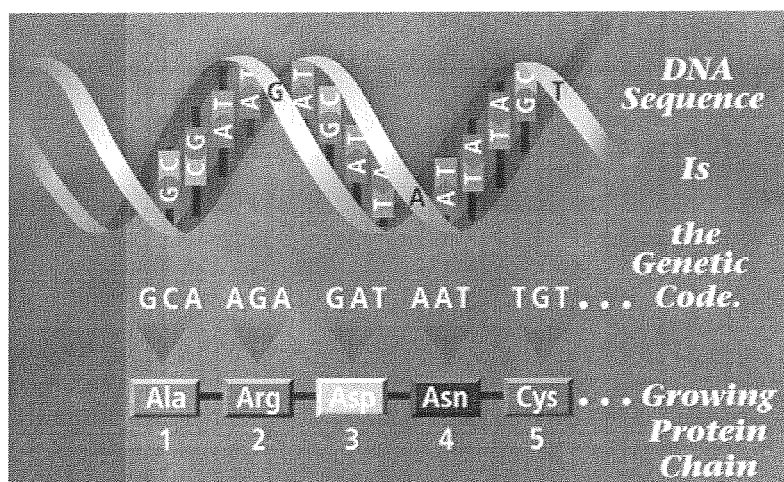


Figura 3 – Síntese de Proteínas [157]

As receitas bioquímicas para a fabricação das moléculas de proteínas estão escritas em código – código genético [141] – na molécula de DNA. A receita para cada molécula de proteína constitui um **gene**. O gene nada mais é que um pedaço de uma longa cadeia de DNA. O conjunto de genes de um indivíduo recebe o nome de **genoma**. Se pudéssemos ler o genoma completo, poderíamos determinar todas as características e funções de um ser vivo, mas isso é extremamente difícil pois o genoma contém uma grande quantidade de informação.

As características de um indivíduo resultam das **proteínas** que formam seu corpo. Como o DNA controla a fabricação dessas proteínas ele acaba por determinar e controlar praticamente todas as nossas características, sejam elas físicas, fisiológicas e até comportamentais [68].

Existem diversos tipos de proteínas entre elas as enzimas (que aceleram e regulam os processos vitais), os hormônios (que levam informações químicas que regulam os ciclos vitais), os anticorpos (que defendem o organismo contra substâncias invasoras), além de proteínas de reserva (albuminas), transporte (hemoglobina) e contração de músculos (actina e miosina).

Proteínas são moléculas grandes (macromoléculas) formadas por aminoácidos. O aminoácido é uma molécula que é sempre formada por dois agrupamentos especiais chamados amina e carboxila que se interligam através de uma ligação peptídica. Existem 20 tipos diferentes de aminoácidos como por exemplo a glicina, a tirosina, a cisteína, a leucina, a prolina, a ácido aspártico, metionina, etc.

Os genes guardam as informações para a síntese de proteínas do nosso corpo que vão desde simples substâncias reagentes a hormônios reguladores de complexos sistemas vitais. Os genes estão localizados nas longas cadeias de DNA, mas nem todo o DNA é formado por eles. Na verdade, apenas 2% do DNA codifica instruções para a síntese de proteínas. O cromossomo 1 é o que mais possui genes (2.968) e o cromossomo Y o que menos possui (231). Grandes seqüências do DNA ainda permanecem um mistério para a ciência e parecem não possuir funções diretas. Outras, suspeitam-se estar associadas à regulagem de onde, quando e em qual quantidade as proteínas são produzidas.

A transformação do DNA nas proteínas acontece em um processo de duas etapas conhecido como transcrição e tradução (Figura 4) [36]. Toda elegância deste processo reside na estrutura de dupla-hélice do DNA que permite que o fio de um lado do DNA, possa identificar por completo o outro lado com perfeita fidelidade.

A primeira etapa, conhecida como **transcrição**, envolve a cópia da seqüência de uma das fitas do DNA na forma de um ácido ribonucléico mensageiro, o mRNA ou ARNm. Esta molécula, que é muito similar ao DNA, também possui 4 tipos de bases nitrogenadas, mas a base uracila (U) substitui a timina (T).

A segunda etapa é conhecida como **tradução**. A informação no mRNA é traduzida com a ajuda de moléculas de RNA de transferência (tRNA), utilizando uma tabela de código genético para determinar a seqüência de aminoácidos. Na tradução, cada grupo de três nucleotídeos, ou **códon**, especifica um aminoácido em particular. Por exemplo, a seqüência AAU indica que o aminoácido asparagina deve ser adicionado à cadeia polipeptídica (proteína).

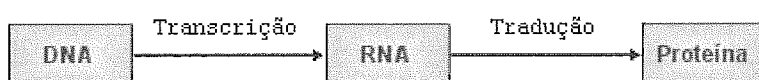


Figura 4 – Transformação do DNA em proteínas

Uma vez que a seqüência de proteína é obtida, inferir a sua estrutura e função representam um grande problema [72]. Proteínas não são moléculas lineares como sugerem o modo de escrita em uma seqüência de “caracteres”, mas sim uma intrincada estrutura tridimensional que é única em cada proteína.

A importância de entender a estrutura da proteína vem de dois fatores: o primeiro é que a função da proteína é dependente da sua estrutura [80]. O segundo fator é que é extremamente trabalhoso de determinar a estrutura de uma proteína experimentalmente. Hoje, muitas seqüências foram determinadas, mas as estruturas tridimensionais permanecem menos exploradas, mesmo sabendo-se que a habilidade de prever exatamente as estruturas (e por conseqüente, as funções) de proteínas pode revolucionar a medicina, a farmacologia, a química e a ecologia.

2.2 Bioinformática

O advento do DNA gerou um crescente aumento nas pesquisas na área de biologia molecular se refletindo diretamente no mundo e na sociedade atual. Hoje, questões como o projeto do genoma humano, testes de paternidade, manipulação do código genético, patentes de genes do DNA e alimentos transgênicos, além de todas as controvérsias que os envolvem, estão presentes diariamente no noticiário dos jornais. Experiências como a da famosa ovelha Dolly [177], o primeiro mamífero a ser clonado, passando pelas vacas Vitória [92] e Vitoriosa [16], os primeiros clones brasileiros, até os anúncios de possíveis clones de seres humanos [6][24], despertaram uma imensa curiosidade e apreensão no mundo inteiro, sendo motivo de acaloradas discussões morais e éticas. Polêmicas à parte, é inegável o potencial das diversas aplicações que a pesquisa do DNA pode gerar, reservando com certeza, um futuro fascinante, promissor e muitas vezes, até mesmo, perturbador.

A par de toda discussão, um outro ramo de pesquisa tem evoluído continuamente e se desenvolvido rapidamente: a **bioinformática**, que é considerada uma área interdisciplinar de pesquisa originada da intensa necessidade da utilização de recursos da ciência da computação aos problemas de biologia [72][93]. Porém ao contrário do que se possa imaginar, a bioinformática não é um ramo de pesquisa tão novo, sendo utilizada gradativamente desde a década de 70. Além disso, os pesquisadores costumam diferenciar o termo bioinformática do termo biologia molecular computacional. Apesar

de ainda não existir um consenso sobre as definições de cada área, como pode ser visto nas discussões da mesa redonda “Perspectivas em Bioinformática” [120] na 1ª Conferência Internacional em Bioinformática e Biologia Computacional 2003 (ICoBiCoBi), tem-se utilizado o termo bioinformática como uma subárea da biologia computacional, que envolve a ciência da computação, a matemática, a estatística e a física, para analisar e comparar seqüências genômicas ou protéicas com a intensa utilização das ferramentas da computação como banco de dados, redes e inteligência artificial. Por outro lado, o termo biologia molecular computacional estaria mais voltado para o desenvolvimento de novos modelos quantitativos e novos paradigmas de computação para os fenômenos biológicos.

Todo o crescimento dessa área se deve principalmente, pois em alguns setores da pesquisa biológica, como a bioquímica e a biologia molecular, o uso da informática é imprescindível para que se possa chegar a resultados precisos a partir dos dados coletados. Esse casamento deu tão certo que a quantidade de informação gerada atualmente é tão grande que está sendo difícil gerenciá-la e manipulá-la por completo, dando surgimento a toda uma coleção de ferramentas com esse único propósito. Esse sucesso deve-se em parte pelo uso da Internet como meio de distribuição dos recursos de bioinformática e da adoção de um princípio conhecido de abstração da informação que simplificou a representação da cadeia tridimensional (3D) de DNA por uma simples seqüência de caracteres repetitivos “A”, “T”, “C” e “G”, representando os quatro possíveis nucleotídeos: adenina, timina, citosina e guanina. O mesmo princípio também se aplica para as seqüências de proteínas com a diferença que existem 20 possíveis aminoácidos.

Dentre as áreas que mais merecem destaque, encontra-se a pesquisa em biologia molecular que tem como principal objetivo determinar os genomas completos de várias espécies, através do seqüenciamento de cadeias de DNA e a identificação da localização e das funções dos genes. No mundo inteiro, diversos centros de pesquisa estão colaborando para executar esta tarefa, destacando-se a espécie humana, no que é conhecido como o **Projeto do Genoma Humano (PGH)** [84] [85]. Em abril de 2003, exatamente 50 anos depois da descoberta de Watson e Crick, os cientistas do PGH anunciaram a conclusão do seqüenciamento dos 3 bilhões de nucleotídeos do DNA humano em um projeto que consumiu US\$ 2,7 bilhões e já tem 13 anos de duração.

Descobriu-se também que existem entre 20.000 e 25.000 genes, distribuídos ao longo de 23 pares de cromossomos, no DNA humano [156].

O Atlas do genoma humano pretende revolucionar práticas médicas e estudos biológicos. Entretanto, a determinação do seqüenciamento completo do genoma não é o final de tudo [47], mas sim o início de outra área de estudo, chamada de **Pesquisa Pós-Genômica**. Se agora possuímos o conhecimento de toda a cadeia de “caracteres” que formam o DNA, é preciso saber quais desses caracteres representam um gene, ou seja, possuem uma função específica no corpo humano e qual é essa função. Atualmente, menos da metade das funções dos genes são conhecidas para qualquer organismo que teve seu genoma seqüenciado.

Através do estudo das estruturas químicas dos genes espera-se entender os dados genéticos replicados pela evolução das gerações e associar os genes as suas funções permitindo dessa forma a determinação de doenças antes mesmo até do nascimento. Essa associação permitirá a fabricação de medicamentos feitos sob medida, tratamentos preventivos e até mesmo a manipulação genética para o conserto de “genes defeituosos”. Em uma experiência muito interessante da área de Engenharia Genética, cientistas eliminaram um gene relacionado à visão das moscas de frutas utilizando técnicas de biologia genética e como resultado obtiveram moscas sem olhos [69]. No futuro, essa mesma técnica poderá ser utilizada para eliminar ou modificar genes relacionados à obesidade, ao mal da Parkson, Síndrome de Dawn, entre outros males.

O mais interessante de tudo, é que a função de um gene, na verdade, nada mais é que a função da proteína que este gene origina, pois é na cadeia de nucleotídeos do DNA que está codificada a informação sobre qual, quanto e como uma determinada proteína deverá ser produzida no nosso corpo e são elas, as responsáveis por muitos dos processos e ciclos vitais que regem um organismo, como os hormônios, as enzimas, os anticorpos, entre outras proteínas. Por sua vez, a função de uma proteína está intimamente ligada com a sua estrutura tridimensional. Portanto, a resolução do trinômio “Função do Gene-Função da Proteína-Estrutura da Proteína” se tornou um dos tópicos de pesquisa mais importantes da atualidade [80].

Neste contexto, três áreas da pesquisa em bioinformática merecem destaque devido a sua importância e a quantidade de pesquisas realizadas e estão diretamente relacionadas a esta dissertação:

- Banco de Dados de Biologia Molecular
- Comparação e Análise de Sequências
- Predição de Estrutura de Proteínas, que é o assunto do estudo de caso desta dissertação.

Banco de Dados de Biologia Molecular

Os bancos de dados de biologia molecular podem ser divididos em diversas características, mas os principais são aqueles que guardam informações de posicionamento da estrutura tridimensional das proteínas, sendo o *Protein Data Bank* [26][27] o seu principal representante, e os que guardam informações de seqüências de DNA ou de proteínas, que variando conforme as espécies pesquisadas e os centros de pesquisa deram origem a diversos bancos, como o *GenBank Sequence Database* [25], o *Annotated Protein Sequence Database (Swiss-Prot)* [19] e o *A. C. Elegans Database (AceDB)* [3]. É importante notar que muitos dos chamados bancos de dados, não utilizam sistemas gerenciadores de banco de dados propriamente ditos, mas sim sistemas de arquivos do sistema operacional ou sistemas de arquivos próprios, ou uma combinação dos três. Na verdade, seria mais sensato chamá-los de fontes de dados ou base de dados, de uma forma geral, mas utilizaremos o termo banco de dados para igualar com a terminologia utilizada na comunidade.

Essa grande heterogeneidade para o armazenamento dos dados nos diversos projetos de pesquisa pode-se configurar em uma grande oportunidade de aprimoramento. A utilização de algum tipo de representação estruturada ou até mesmo semi-estruturada, que possua mecanismos de restrições e validações, como os arquivos XML, podem melhorar a utilização desses dados e trazer maior semântica aos mesmos.

O principal banco de dados de seqüências de DNA do mundo é o *Genbank*. Mantido pelo *National Center for Biotechnology Information* (NCBI) [112], ele foi estabelecido em 1978 como um repositório central para os dados biológicos. O tamanho do banco de dados tem dobrado aproximadamente a cada 15 meses. Mais de 140.000 diferentes espécies são representadas no *GenBank* e novas espécies são adicionadas a

taxa de 1700 por mês. Registros de humanos constituem 26% do total dessas seqüências [25].

O sistema é mantido como uma combinação de arquivos “*flat*”, banco de dados relacionais e arquivos no formato Abstract Syntax Notation One (ASN.1), uma sintaxe para definição de estruturas de dados desenvolvida pela indústria de telecomunicações. O NCBI construiu o GenBank, inicialmente através da submissão direta de dados de seqüência pelos autores, mas atualmente os dados de outros bancos públicos de seqüências como o EMBL [139] e o DDBJ [115] são incluídos automaticamente no GenBank, através da troca de dados diária entre esses três sistemas. Cada entrada no *GenBank* inclui uma descrição concisa da seqüência, como mostrado parcialmente na Figura 5.

BASE COUNT	213 a	58 c	63 g	150 t		
ORIGIN						
1	attattaaaa	ggttaaacta	cattaaaaaca	taaagagaga	ggaattgcta	tgacagtatt
61	tgtagatcat	aaaattgaat	acatgagttt	agaagatgat	gctgaacttt	taaaaacaat
121	ggcacatcct	atgcgtttaa	aaatagtcaa	tgaactttac	aaacataaag	cattaaatgt
181	aacgcaaatc	attcaaatct	taaaactacc	acaatcaact	gtatcccage	atttatgtaa
241	aatgagagga	aaagttttaa	aaagaaatcg	acaaggttta	gagatatact	atagcattaa
301	taatccaaaa	gttgaaggga	ttattaagtt	gttaaaccct	atccaataga	ttttatataa
361	ttacctccta	ttttaagggt	ttaatagaaa	taaaaatcct	atttattaca	ataagaaagc
421	ttatattttc	atttattaaa	caaagggaaa	aagtaataaa	aatcttataa	aaaagacgct

Figura 5 – Seqüência de bases de nucleotídeos

O *Protein Data Bank* (PDB) é um repositório universal de dados de estruturas de macromoléculas biológicas submetidas pelos pesquisadores de todo o mundo a partir da execução de métodos experimentais de raios-X (cristalografia) e ressonância magnética nuclear (NMR, *Nuclear Magnetic Resonance*) e também de modelos teóricos computacionais. O PDB foi estabelecido em 1971 e desde então teve um enorme crescimento, contando em novembro de 2004 com 27.900 estruturas e expectativa de triplicar de tamanho até 2008 [172].

A melhora da qualidade das tecnologias de cristalografia, a adição de novos métodos como a ressonância magnética nuclear e o advento da Internet permitiram o crescimento e o estabelecimento do mais importante centro de referência para o estudo de proteínas. Um exemplo de um trecho de uma seqüência no PDB é mostrado na Figura 6. As linhas que começam com ATOM representam as coordenadas tridimensionais de cada átomo da proteína, onde as três últimas colunas mostram os valores para as coordenadas X, Y e Z.

HEADER	B-DNA						
COMPND	G-C	B-DNA	BASE	PAIR			
AUTHOR	GENERATED BY GLACTONE						
SEQRES	1	A	1	G			
SEQRES	1	B	1	C			
ATOM	1	P	G A	1	-6.620	6.196	2.089
ATOM	2	OXT	G A	1	-6.904	7.627	1.869
ATOM	3	O2P	G A	1	-7.438	5.244	1.299
ATOM	4	O5'	G A	1	-5.074	5.900	1.839
ATOM	5	C5'	G A	1	-4.102	6.424	2.779
ATOM	6	C4'	G A	1	-2.830	6.792	2.049
ATOM	7	O4'	G A	1	-2.044	5.576	1.839
ATOM	8	C3'	G A	1	-2.997	7.378	0.649
ATOM	8	C3'	G A	1	-2.997	7.378	0.649

Figura 6 - Exemplo de um trecho de um arquivo do PDB

Por fim, o *Swiss-Prot* é um banco de seqüências de proteínas que possui um alto grau de verificação dos dados e um grande número de anotações (descrições mais especializadas). Esse tipo de banco é utilizado quando há a necessidade de trabalhar com dados mais confiáveis sobre as seqüências.

Comparação e Análise de Seqüências

A comparação de seqüências biológicas [128] é uma das tarefas mais importantes da biologia molecular e consiste em encontrar trechos semelhantes em duas ou mais seqüências dadas. A busca por essas semelhanças é utilizada em uma vasta gama de problemas distintos entre eles quando um gene é seqüenciado por dois laboratórios diferentes e deseja-se comparar os resultados, ou quando duas seqüências consecutivas precisam ser agrupadas em uma única através da semelhança do prefixo de uma com o sufixo da outra.

	A	-	C	G	T	G	T	T	C	G	A	T	G	C	T	A	-	C	T	A
	A	T	C	G	T	T	A	T	C	-	A	T	T	C	T	T	G	C	T	A
placar	1-2	1	1	1	1-1	1	1-2	1	1-1	1	1-1-2	1	1	1	1	1	1	1	1	= 5

Figura 7 - Exemplo de alinhamento de seqüências

O uso de computadores é largamente utilizado nesta tarefa e diversos algoritmos foram propostos para resolver este problema. Esses algoritmos utilizam a idéia de **alinhamento** entre duas seqüências, que pode ser visto como a colocação da primeira base (ou letra) de uma seqüência lado a lado à primeira base da outra seqüência e assim por diante. Como as duas seqüências não precisam ser do mesmo tamanho, pode haver a inserção de buracos em pontos arbitrários de modo que elas fiquem do mesmo tamanho. Os buracos podem também ser colocados no início ou fim das seqüências, desde que um buraco não esteja alinhado com outro buraco na outra seqüência, conforme é mostrado na Figura 7.

Os algoritmos tentam buscar o melhor alinhamento através de um esquema que distribui pontos para cada base alinhada, conforme a igualdade ou diferença entre as bases ou ainda a inserção de buracos. Assim, seria dado 1 ponto para duas bases iguais, -1 para duas diferentes e -2 se houver uma base e um buraco. A pontuação total do alinhamento será a soma dos pontos de cada base. O alinhamento ótimo é aquele que possui pontuação máxima e é chamado de **similaridade** entre duas seqüências. Diferentes algoritmos podem utilizar esquemas de pontuação diferentes, além de ser possível manipular os pontos dados e os pontos retirados, que são chamados de penalidades.

Algoritmos de Programação Dinâmica computam todos os alinhamentos possíveis para escolher aquele que tem a maior pontuação, ou seja, possuem similaridade ótima. Os dois algoritmos que formam a base para a maioria dos métodos são Needleman-Wunsch [113] e Smith-Waterman [136]. O primeiro utiliza alinhamento global enquanto que o último utiliza alinhamento local. A complexidade de tempo desses algoritmos deriva da matriz de comparação que é proporcional ao produto do tamanho de duas seqüências. Então, a complexidade de tempo para obter-se o melhor placar é da ordem de $O(nm)$, onde N e M são os tamanhos de duas seqüências. Esses algoritmos aplicados a buscas de banco de dados são impraticáveis. O tempo para procurar em um banco de K seqüências é da ordem de $K(nm)$, onde K é tipicamente do tamanho de 10^5 . Apesar de serem mais caros computacionalmente, são mais precisos em encontrar acertos significativos para um dado esquema de pontuação. Esses métodos devem ser escolhidos quando uma comparação rigorosa é requerida.

Algoritmos Heurísticos, como o BLAST [8] e FASTA [174], operam com a heurística que cada base (ou letra) de ambas as seqüências não necessitam de serem comparados uma a uma para detectar altos placares de alinhamento. Ambos algoritmos primeiramente identificam segmentos pequenos e altamente similares que são então expandidos. Eles assumem, principalmente, que qualquer alinhamento significativo cercam um ou mais desses segmentos. A complexidade desses algoritmos permanece na ordem de $O(nm)$, mas o número de computações baseadas nas comparações bases a bases é altamente reduzida. Esses dois algoritmos não garantem o encontro de alinhamentos ótimos, mas provaram serem muito efetivos para a comparação de seqüências, além de serem muito mais rápidos.

A comparação entre duas seqüências pode ser estendida para a **busca de seqüências por todo um banco de dados**. Essa busca permite a comparação entre seqüências de várias espécies para obtenção de semelhanças entre elas ou a comparação de um mesmo gene de diferentes indivíduos da mesma espécie para localizar diferenças, que podem indicar doenças hereditárias.

Predição de Estrutura de Proteínas

A predição da estrutura de proteínas pode ser definida como a necessidade de se conhecer as estruturas tridimensionais das proteínas relacionadas a um gene, e dessa forma, determinar sua função. A partir da seqüência de DNA, é relativamente fácil descobrir as seqüências de aminoácidos resultantes, seguindo o mapeamento da transcrição-tradução mostrada anteriormente. Entretanto, descobrir a estrutura tridimensional da proteína a partir dessas seqüências é um dos objetivos principais da biologia atualmente, e que muitos consideram como o segundo código genético a ser explorado.

Muitos fatores contribuem para dificultar a predição de estrutura de proteínas. Os 20 valores possíveis para diferentes aminoácidos se juntam em uma seqüência e formam aproximadamente 1000 diferentes estruturas principais, chamadas de dobras (*folds*), cada uma com dezenas ou centenas de variações. As forças físicas que governam as interações entre centenas ou milhares de aminoácidos determinam a estrutura da proteína. Ainda não são totalmente conhecidos os detalhes dessas interações, e mesmo que isso acontecesse, computá-los seria extremamente trabalhoso. Esse tópico de pesquisa é tão desafiante que existe inclusive uma competição, chamada *Critical Assessment Structure Prediction (CASP)* [39] que premia técnicas inovadoras na área.

A maioria dos dados de estruturas tridimensionais conhecidos foi obtida a partir de três métodos: raios-X cristalográficos (mais de 80%), ressonância magnética de solução nuclear (15%) ou modelagem teórica (5%). Os dois primeiros são métodos experimentais (em laboratórios) e descrevem a estrutura da molécula por meio de técnicas de raios-X ou ressonância magnética, no estado da qual as medidas foram feitas, isto é, a medição depende da temperatura, pressão, pH, tipo de solventes utilizados, etc.

Estruturas obtidas de modelos teóricos (computacionalmente) tendem a ser menos acuradas e baseiam-se em dois métodos [21]. O primeiro inclui modelagem por comparação, ou **homologia**, que se baseia em comparações entre a estrutura tridimensional da proteína estudada com as proteínas cujas funções já são conhecidas. A existência de alguma similaridade entre essas proteínas permite inferir algum conhecimento sobre a função da proteína estudada com base na proteína comparada. Resumidamente, este método é baseado em quatro etapas: encontrar estruturas conhecidas relacionadas à sequência a ser modelada; alinhar a sequência com essas estruturas; construir um modelo; e avaliar este modelo. As estruturas para o modelo podem ser encontradas por algoritmos de comparação de sequência, como o BLAST, ou por algoritmos de execução em fileiras de estrutura-sequência (*threading*) [182].

O segundo método, prediz a estrutura de uma sequência isolada, sem se preocupar com similaridades, mas sim buscando a estrutura que minimiza a energia livre de uma proteína, no método conhecido como *ab initio* ou *de novo*. Neste método, não é necessário o conhecimento de pelo menos uma estrutura da família de estruturas que se está pesquisando. Ele assume que o estado nativo de uma proteína é aquele que minimiza a energia livre global. Os dois passos são a busca de uma larga escala de espaços conformacionais de estruturas ternárias e a função de energia livre utilizada para avaliá-los.

A acurácia de modelos comparativos é maior que os métodos de energia livre, mas em ambos os casos, dependendo das circunstâncias, é possível uma boa predição. Apesar de a maioria das estruturas terem sido descobertas experimentalmente, acredita-se que apenas com a evolução da modelagem de proteínas poderemos encontrar as estruturas da maioria dessas moléculas.

2.3 Principais Desafios

Mesmo com todos os benefícios que se esperam, os projetos de biologia molecular tornaram-se um grande desafio. O uso da tecnologia da informação permitiu que diversas análises e descobertas fossem efetuadas, das quais seriam extremamente lentas de outras maneiras, mas, em contrapartida, gerou um aumento exponencial da quantidade de informação produzida.

Hoje, podemos afirmar, que o nível de dependência dos recursos computacionais pelos cientistas que estudam as seqüências de nucleotídeos, os genes e as proteínas, é muito grande, e vai desde programas para análises individuais de pequenas seqüências de DNA, até ferramentas de busca e predição que analisam todo um genoma para a descoberta de novos medicamentos. Dessa forma, existe uma enorme demanda por ferramentas eficientes e sofisticadas que auxiliem nas tarefas de coleta, representação, organização, armazenamento e análise de todos esses dados [81][100][133].

Atualmente, os principais desafios da área de bioinformática podem ser resumidos em três tópicos:

- Grande heterogeneidade de bancos de dados e aplicações
- Necessidade de integração entre diversos bancos de dados
- Definição e Execução de experimentos através da composição de aplicações científicas, os chamados experimentos *in silico*, a partir da interoperabilidade entre os diversos programas científicos, que é o foco desta dissertação.

Grande Heterogeneidade de Banco de Dados e Aplicações

A grande **heterogeneidade** entre as bases de dados é uma das características mais marcantes da área de bioinformática. Devido à imensa quantidade e variedade, os diversos sistemas são projetados conforme as necessidades de cada centro de pesquisa gerando representações específicas em cada um deles. Além de esquemas diferentes para a mesma informação, é comum encontrar formatos de dados usados com diferentes representações. Os aminoácidos que formam uma proteína, por exemplo, são ambigualmente representados com uma letra só (“A” para o aminoácido alanina, “L” para a leucina) ou como três letras (“ALA” e “LEU”, respectivamente). Essas bases também são implementadas de diferentes maneiras, desde repositórios de arquivos com textos semi-estruturados, passando por coleções de páginas *Webs* até o uso de sistemas gerenciadores de banco de dados.

Atualmente, existem diversos bancos de dados (ou fontes de dados em geral) de seqüências de biologia molecular como mostrou Baxevanis [23] em uma extensa lista. Esses bancos cobrem os mais diversos aspectos dos mais diferentes organismos destacando-se os bancos de dados de seqüência de DNA como GenBank [25], AceDB [3], EMBL [139] e DDBJ [115] e; os de seqüência de proteínas como Swiss-Prot [19],

PIR [180] e TREMBL [19]; os de estruturas de proteínas como PDB [26][27] e DSSP [57]; os de dados biológicos relativos a organismos específicos como os humanos, mamíferos e bactérias [150]; os que se concentram em uma função biológica específica [20]; e ainda, os que registram mutações em um gene ou grupo de genes [4].

A grande heterogeneidade também é uma das características dos programas e aplicações utilizados pelos biólogos e cientistas. É muito comum a mesma solução para um problema ou até um mesmo algoritmo, possuírem diversas implementações em centros de pesquisas diferentes. Muitas vezes, isso acontece, pois cada centro de pesquisa desenvolve suas aplicações individualmente sem se preocupar com questões como interoperabilidade e escalabilidade.

Dessa forma, as diversas aplicações que se complementam e deveriam interagir entre si, possuem grande dificuldade em se comunicarem, necessitando de um esforço muito grande por parte dos biólogos, para manualmente, pegarem a saída de dados de um programa, transformá-la e convertê-la na entrada do outro. Considerando que a interface entre esses programas normalmente são bem diferentes, como um programa executável ou uma página Web, esse processo manual piora ainda mais. Uma interface mais padronizada entre essas diversas aplicações e uma maneira mais razoável para invocá-los é extremamente necessária, e neste caso, a tecnologia de serviços Web surge como uma grande alternativa.

Outras iniciativas para tentar diminuir o impacto deste problema caminharam na direção do desenvolvimento de bibliotecas reutilizáveis de código aberto englobando diversos recursos e ferramentas utilizadas na bioinformática. Assim, tarefas comuns como representação de uma seqüência de DNA ou proteína, métodos para importar e exportar dados entre fontes de dados, conversão entre diferentes formatos e *parsers* de arquivos já estão implementados em bibliotecas públicas como bioPerl [146], bioJava [145] e bioPython [147] prontas para serem utilizadas no desenvolvimento de novas ferramentas. Hoje esses grupos estão organizados em uma entidade comum chamada *Open Bioinformatics Foundation* [152] que pretende dar apoio centralizado à programação de código aberto em bioinformática. Embora iniciativas deste tipo sejam de vital importância, a real utilização dessas bibliotecas ainda não está totalmente disseminada entre os diversos centros de pesquisa ao redor do mundo.

Necessidade de Integração entre Diversos Bancos de Dados

Outro grande desafio, como notado em [46][125], se refere à **integração** de centenas, se não milhares de programas e fontes de dados de bioinformática. Apenas através da integração desses recursos, será possível o compartilhamento de diferentes informações complementares e relacionadas, possibilitando a execução de comparações e análises a fim de atingir novas descobertas científicas. Dessa maneira, ferramentas que fornecessem acesso único aos diversos recursos, além da execução de consultas entre várias fontes de dados são extremamente valiosas.

Entretanto, como mostraram Paton e Goble [122] e Davidson *et al.* [56], a integração de bases de dados de biologia molecular não constitui uma tarefa trivial. Além dos problemas tradicionais de integração como heterogeneidade do ambiente, autonomia das bases individuais, distribuição e consistência, a área de bioinformática possui requisitos particulares como a falta de um esquema global dos dados, a necessidade do uso de texto puro nas consultas, a inexistência, em muitos casos, de um sistema gerenciador de banco de dados por baixo das fontes de dados, além do amplo uso de um campo de texto livre, conhecido como anotações [109], utilizado para guardar desde informações relevantes provenientes da análise individual do biólogo, até os dados sobre as condições do experimento (quantidade de material, luminosidade, pH, presença de alguma substância, etc).

A abordagem mais primitiva de integração baseia-se na navegação hipertexto, onde o usuário pode navegar através de ligações existentes entre registros de diferentes fontes de dados (Entrez [61]) ou através de um sistema de navegação que criam esses *links* (SRS [62], LinkDB [67]) para integrar as diversas fontes. Essa alternativa é eficaz quando o próprio usuário está analisando um determinado dado e consegue através dos *links* ter acesso aos dados relacionados, mas deixa muito a desejar quando necessitamos de consultas um pouco mais complexas que inter-relacione estes dados, pois junções não são definidas.

Uma abordagem mais elaborada trata diferentes bancos de dados como se fossem um único sistema, apesar de não estarem fisicamente integrados. Um mecanismo de consulta fica encarregado da tarefa de acessar os diversos bancos de dados, dividir as consultas e enviá-las para cada um dos bancos. Após executar estas consultas, o

resultado é centralizado em um mediador, que fica responsável de juntar as respostas individuais em um único resultado final. Esse mediador pode ser implementado baseado em uma linguagem específica que permite representar os tipos de dados utilizados na biologia, em um nível maior de abstração. O mediador ficaria responsável também por fazer o mapeamento desses tipos complexos para os tipos específicos de cada base de dados através de tradutores (*wrappers*). Essa abordagem baseada em mediadores, também conhecida como banco de dados múltiplos (*multidatabases*), é utilizada por sistemas como K2/Kleisli [55] e TAMBIS [123].

A maior dificuldade dessa abordagem reside em manter atualizado o esquema global do sistema de forma a refletir as constantes modificações nos esquemas individuais das fontes de dados. Outra desvantagem, refere-se a necessidade de criar novos tradutores para cada nova base que se deseje adicionar ao sistema. Além disso, essas implementações tendem a apresentar resultados mais lentos, quando são necessárias várias consultas simultâneas através da Internet.

Outra abordagem de integração está mais voltada para o uso *warehouses* que pode ser visto como um banco de dados central onde os dados de diversas fontes são colocados fisicamente juntos. Dessa maneira, um esquema global pode ser utilizado para materializar os dados em um repositório central do sistema e as consultas são executadas diretamente no *warehouse*. Sistemas como GUS [55], GIMS [50], GenAlg [79] e DataFoundry [51] adotam essa abordagem que normalmente apresentam um bom desempenho nas consultas, mas apresentam dificuldades na atualização dos repositórios centrais (*warehouses*) para refletir as mudanças das fontes individuais.

Diversas outras iniciativas existem na literatura [134] para o problema de integração de bancos de dados de bioinformática. A utilização de ontologias [77][78] para a descrição e classificação mais semântica desses dados de bioinformática tem cada vez mais surgido na literatura [14][179]. Entretanto, por já existirem muitas iniciativas nesta área, este tópico não é o objetivo desta dissertação, já que iremos focar mais na integração de programas através da utilização de workflows.

Para o caso dos programas, existem ainda iniciativas onde vários programas utilizados na área de bioinformática, são disponibilizados em um único pacote, como é o caso do EMBOSS [149]. Também foram desenvolvidos diversos sistemas voltados à

integração e padronização de diferentes recursos de bioinformática como: o DAS (*Distributed Annotation System*) [54], um sistema para integrar anotações de seqüências de dados de genoma distribuídos e o I3C (*Interoperable Informatics Infrastructure Consortium*) [89], que é um grupo que promove a adoção de padronização para facilitar o intercambio de dados e a interoperabilidade de programas.

Apesar das virtudes dessas iniciativas, acreditamos que a adoção da tecnologia de serviços Web possa constituir um elemento incentivador para a criação de um padrão de fato para a interoperabilidade entre as diversas aplicações, permitindo mais facilmente a comunicação entre elas.

Definição e Execução de Experimentos através da Composição de Aplicações Científicas

Nas ciências como física, biologia, química entre outras, a busca por novos conhecimentos é executada na forma de **experimentos científicos**, ou seja, a execução sistemática de operações que visam validar e verificar algum resultado. Nessa busca, os cientistas normalmente seguem duas abordagens para conduzir seus experimentos: a abordagem baseada em hipótese e a abordagem baseada em dados, como mostrou Chen [45].

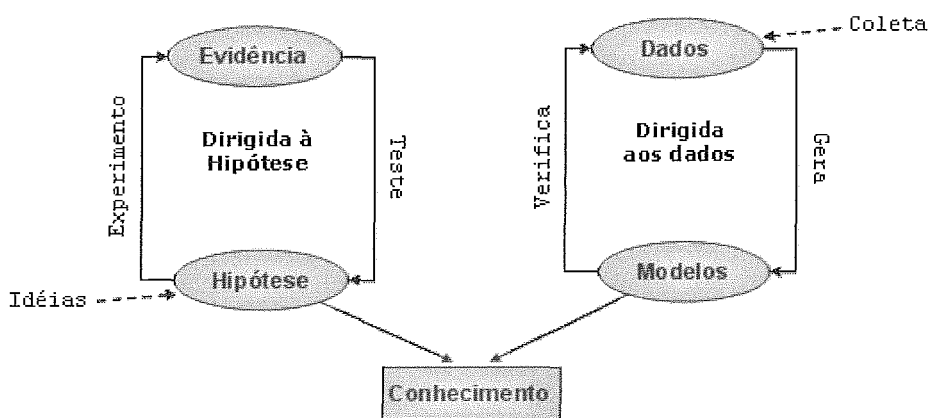


Figura 8 - Metodologia baseada em hipótese versus baseada em dados, livremente adaptada de [45]

Muitos dos biólogos tradicionais seguem a abordagem dirigida à hipótese. Eles acreditam que uma hipótese, surgida de idéias baseadas na experiência pessoal e em observações de experimentos, é o principal caminho para a descoberta do conhecimento científico. Após a definição da hipótese, o biólogo produz evidências pela execução de experimentos que testam suas hipóteses. A evidência experimental pode fortalecer ou

enfraquecer a hipótese inicial gerando os ajustes necessários e uma nova repetição do ciclo de “experimentação e teste” no processo conhecido como exploração científica (Figura 8). Somente hipóteses que permanecem estáveis sobre exaustivos ciclos de examinação poderão ser chamadas de conhecimento.

Os cientistas computacionais tendem a seguir a abordagem baseada em dados. Nessa abordagem o ciclo inicia com grandes quantidades de dados, geralmente produzidos por processos de captura de dados de alta capacidade como o seqüenciamento de DNA. Então, os cientistas executam processos intensivos sobre estes dados para a produção de modelos. Esses modelos descrevem estruturas biológicas em alto nível e podem se tornar conhecimento se puderem ser utilizados diversas vezes em novos dados (Figura 8). Essa abordagem também é conhecida como análise de dados em larga escala (processo guiado) ou *data mining* (processo não guiado).

Quando o processo de experimentação científica através dessas duas abordagens é executado pela utilização intensiva de recursos computacionais, convencionou-se chamar de experimento *in silico*, uma analogia ao termo *in vitro*. Enquanto o último tem origem em experimentos em laboratórios utilizando tubos de ensaio (de vidro ou *in vitro*), o termo *in silico* nasceu justamente para representar os experimentos através do uso de computadores, que possuem como matéria prima o elemento químico silício (*in silico*).

Os experimentos *in silico* portanto, visam representar cada uma das atividades anteriormente feitas em laboratório, por programas de computadores que devem ser executados em um determinado fluxo, onde normalmente a saída de um dos programas servirá de entrada para o próximo na cadeia de execução. Dessa forma, efetua-se uma composição dos diversos programas disponíveis de forma que o conjunto deles satisfaça a um objetivo maior.

Basicamente o ciclo de vida de um experimento *in silico* se dá em quatro etapas, como mostra a Figura 9.

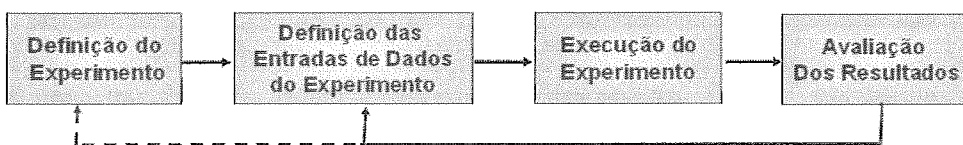


Figura 9 – Etapas do Experimento *in Silico*

Definição do Experimento: Nesta etapa, o cientista é responsável por descrever a seqüência das diversas atividades (os programas) que fazem parte do experimento. Será necessário indicar qual programa inicia o experimento, quais são os programas subsequentes, qual a ordem de execução, quais possíveis desvios do fluxo de execução podem ocorrer e qual será a atividade final do experimento.

Atualmente, essas escolhas são feitas pelos biólogos guiados pela sua experiência própria, sem o auxílio de nenhum mecanismo de busca e classificação dos programas disponíveis. Muitas vezes, nem sempre o melhor programa é escolhido, minimizando ou até mesmo inviabilizando o sucesso do experimento. Uma vez definido o fluxo do experimento, existe uma grande dificuldade de compartilhá-lo para outros cientistas avaliarem ou reutilizarem esta definição em projetos de colaboração entre centros de pesquisa, ou por não haver nada especificado (apenas na cabeça do cientista) ou pela definição estar embutida em alguma linguagem de programação ou de *scripts*. Pelas mesmas razões, manutenções e evoluções no experimento são igualmente difíceis. Entretanto, essas grandes dificuldades na definição dos experimentos incentivam a adoção de workflows científicos para tratar essas questões. Na seção 3.2 veremos como os workflows são próprios para lidar com questões como essas e na seção 3.4 veremos diversas iniciativas que estão utilizando-os para resolver este tipo de problema.

Definição das Entradas de Dados do Experimento: Uma vez definido o experimento na etapa anterior, dar-se-á início à fase de “experimentação e teste” onde o experimento será executado repetidas vezes. Na etapa de definição de entradas de dados serão estabelecidos os valores que serão utilizados na próxima etapa.

Entretanto, da mesma forma que acontece na definição do experimento, a escolha dos dados de entrada adequados, avaliando as diversas alternativas existentes, ocorre sem nenhum mecanismo de suporte aos biólogos. Experiências anteriores não podem ser consultadas e também existe uma grande dificuldade na classificação de uma hierarquia semântica entre os diversos dados e aplicações. Ontologias e metamodelos podem ser utilizados para auxiliar neste processo como veremos na seção 3.4.

Execução do Experimento: Nesta etapa, todos os programas envolvidos no experimento serão executados conforme a seqüência estabelecida na etapa de definição e utilizando os dados definidos na etapa anterior. Pode-se considerar que cada execução

do experimento com dados de entrada diferenciados é uma instância do experimento definido na primeira etapa.

A primeira grande dificuldade refere-se à invocação dos diversos programas que constituem o experimento. Se eles estão localmente instalados na máquina do cientista o problema é minimizado, mas o experimento ficará restrito a ele. Se ele está disponível em uma arquitetura de rede, ou até mesmo através de uma página Web, a problemática se inverte: é mais fácil de acessá-lo, porém mais difícil de invocá-lo. Como já comentado nesta dissertação, acreditamos que a tecnologia de serviços Web é um fator determinante para tratar este tipo de problema, pois possibilita uma maior interoperabilidade entre as diversas aplicações científicas ao mesmo tempo em que preserva a privacidade.

A segunda dificuldade se refere à re-executar apenas um trecho do experimento, o que normalmente ocorre, quando os primeiros passos já tiveram resultados satisfatórios, mas o final do experimento precisa de execuções adicionais, alterando os parâmetros dos programas, para chegar ao resultado desejado. Essas re-execuções parciais são extremamente necessárias aos cientistas para efetuar o ciclo de “experimentação e teste” comentado anteriormente. Para evitar que o cientista execute todo o experimento novamente ou tenha que executar o trecho final manualmente, o registro da execução de programas e workflows deverá ser efetuado, conforme veremos a seguir.

Avaliação dos Resultados: Diversas instâncias do experimento serão executadas seguidas vezes e a cada execução o cientista ficará encarregado de analisar os resultados gerados, efetuando os ajustes necessários baseados na sua percepção conforme a sua pesquisa específica. A cada novo ajuste, um novo ciclo de experimentação será executado. Menos freqüentemente, a avaliação dos resultados pode dar origem à modificação da definição do experimento, adicionando ou removendo programas, por exemplo.

Entretanto para que isso ocorra de maneira natural, o registro das execuções dos programas e do próprio experimento devem ser armazenados em algum mecanismo de persistência para possibilitar futuras consultas. Por registro da execução de um programa entendemos o armazenamento de seus dados e parâmetros de entrada, seus dados resultantes e meta-informações como quem executou, quando e como. Essa

identificação sobre a procedência e transformações executadas sobre os dados é conhecida como *data provenance* [34][35] e constitui-se atualmente em um grande tópico de pesquisa. A possibilidade de descrever e armazenar uma transformação (ou derivação) pode dizer ao biólogo mais informação sobre um dado, permitindo descobrir conexões e relacionamentos com dados já existentes ou se uma determinada análise já foi feita. Isso tudo seria feito sem nenhum esforço de documentação por parte do biólogo, mas sim pelo sistema que automaticamente registre as atividades dos cientistas.

Workflows e Serviços Web na Bioinformática

“Homens e mulheres padronizados, em grupos uniformes. Todo o pessoal de uma pequena usina constituído pelos produtos de um único ovo bakanovskizado. Noventa e seis gêmeos idênticos fazendo funcionar noventa e seis máquinas idênticas. (...) O princípio da produção em série aplicado enfim a biologia”.

Aldus Huxley - Admirável Mundo Novo

Como apresentado no Capítulo 1, uma das contribuições desta dissertação é a utilização de **serviços Web** e **workflows** na implementação de experimentos *in silico*. Neste capítulo, veremos mais detalhadamente esses dois conceitos, apresentando seus objetivos, seu funcionamento, suas principais características e o papel que desempenham na bioinformática. Na última seção, é feita uma análise de diversos trabalhos relacionados que possuem, em maior ou menor grau, objetivos semelhantes ao desta dissertação. Para isso, foi definido um conjunto de dez características importantes para soluções de workflows científicos para bioinformática, chamado de características 10+C, que foi utilizado de base para a avaliação e comparação desses trabalhos.

3.1 Serviços Web

Um serviço Web [160] é um programa descrito através de uma interface padronizada que permite a disponibilização de um serviço em uma rede de computadores, geralmente a Internet. Uma vez descrito na forma padrão, o serviço se torna um componente de software totalmente reutilizável, permitindo a comunicação e a interoperabilidade entre aplicações heterogêneas.

Essa comunicação baseada em padrões, permite que qualquer aplicação, utilizando o protocolo comum HTTP [151] da Web, acesse e utilize serviços sem saber detalhes da implementação do mesmo. Dessa forma, os serviços Web se tornaram para as interações entre programas (B2B, *business-to-business*) o que a Web é para as interações entre pessoas e aplicações (B2C, *business-to-consumer*).

Diferente das tecnologias de componentes atuais, os serviços Web não são acessados através de protocolos de modelos de objetos específicos, como DCOM [107], RMI [140] ou IIOP [114]. Ao invés disso, os serviços são descritos e acessados utilizando uma notação padronizada de XML [164] que cobre todos os detalhes necessários para interagir com o serviço, descrevendo as funcionalidades, a localização, o modo de invocação e os protocolos utilizados para isso. O tripé XML que mantém a arquitetura de implementação dos serviços Web está centralizada em três elementos: WSDL [161] como descritor dos serviços, SOAP [162] como protocolo de comunicação e UDDI [154] para catálogo dos serviços, detalhados mais a frente neste seção.

Uma das grandes vantagens da utilização de XML é que ela é extensível e uma boa alternativa para a descrição de dados semi-estruturados. Dessa forma, a informação fica organizada no arquivo de forma hierárquica, pode ser estendida conforme o domínio do problema e ainda pode ter o seu conteúdo validado segundo algum documento que contenha a definição do esquema da aplicação (utilizando um arquivo DTD [166], por exemplo). XML é um padrão aberto promovido pelo W3C (*World Wide Web Consortium*) [167], que é a organização de padrões da Web.

A simplicidade desta arquitetura, baseada nos padrões XML, é que alavanca todo o potencial deste modelo, ao atacar um dos principais pontos fracos dos sistemas distribuídos: a interoperabilidade entre as aplicações, independente de sistemas operacionais, linguagens de programação, modelos e ambientes proprietários.

Os serviços podem ser implementados usando qualquer linguagem de programação (Java, C++, Visual Basic, etc.), e em qualquer plataforma. Um cliente de um serviço Web também pode ser escrito usando qualquer linguagem e em qualquer plataforma. Por exemplo, um cliente escrito em Delphi na plataforma Windows pode utilizar serviços de um serviço Web escrito em Java na plataforma Linux. O cliente não precisa se preocupar como o serviço foi implementado. Ou seja, os serviços Web

dependem da habilidade das partes para se comunicar mesmo que usem plataformas diferentes.

Um serviço Web também poderá ser facilmente localizado na Internet, uma vez estando publicado na grande rede e registrado em um catálogo UDDI. A comunicação feita via HTTP também garante que a comunicação entre as aplicações acontecerá mesmo com a presença de *firewalls* ou *proxies*, já que a porta default 80 não é normalmente bloqueada.

A arquitetura de serviços Web [95] é baseada sobre a interação entre três entidades: provedor de serviços (*service provider*), o consumidor de serviços ou cliente (*service requestor*) e o registro de serviços (*service registry*). O provedor de serviços é o que cria e desenvolve o serviço Web que serve para expor alguma funcionalidade de negócio da sua organização para a invocação por outros usuários externos. O consumidor será qualquer usuário que deseja utilizar algum serviço Web. O registro de serviços é um diretório central onde o provedor de serviços possa cadastrar e descrever seus serviços, e onde o consumidor possa procurar pelo serviço desejado.

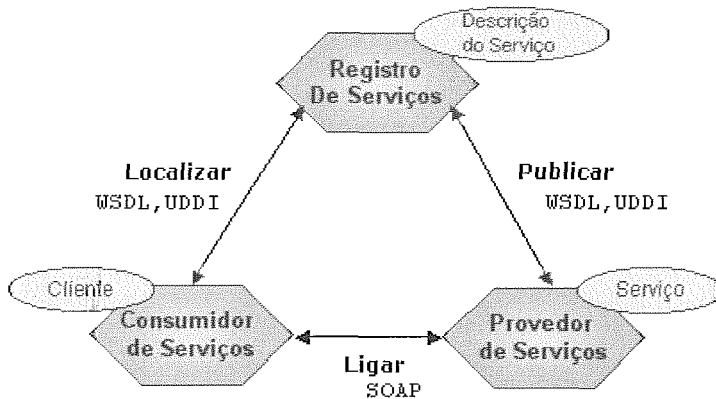


Figura 10 - Arquitetura dos serviços Web.

As três entidades interagem entre si através das operações de publicar (*publish*), localizar (*find*) e ligar (*bind*). O provedor informa ao registro a existência de um serviço Web, usando a interface de publicação do registro, para tornar o serviço disponível aos clientes. A informação publicada descreve o serviço e especifica o local onde se encontra. O consumidor consulta o registro para localizar um serviço Web publicado. Com essa informação, o consumidor faz uma ligação da sua aplicação com o provedor do serviço para poder invocá-lo e interagir com a sua implementação. A Figura 10 ilustra essas operações, os componentes que as provêm e suas interações.

A interface de um serviço Web é descrita em uma linguagem chamada *Web Service Description Language* (WSDL). É a descrição WSDL que habilita qualquer programa a entender como interagir com o serviço Web, pois ela possui as informações sobre quais operações um serviço possui, quais os tipos de entrada e saída de cada operação e qual a localização e o tipo de protocolo utilizado para invocar o serviço. O uso do WSDL na arquitetura de serviços Web convencionalmente divide a descrição do serviço em duas partes: a interface do serviço e a implementação do serviço.

A definição da interface do serviço é uma descrição abstrata que pode ser referenciada e utilizada por múltiplos serviços. É análoga à definição de uma classe abstrata em uma linguagem de programação orientada a objetos. A interface do serviço é composta dos seguintes elementos WSDL: *WSDL:binding*, *WSDL:portType*, *WSDL:message* e *WSDL:type* (Figura 11). O elemento *WSDL:portType* define as operações do serviço, isto é, quais mensagens XML de entrada e saída serão utilizadas. Pense nesse elemento como a assinatura de um método em uma linguagem de programação. O elemento *WSDL:message* especifica o tipo de dado que irá aparecer nas operações de entrada e saída, ou seja, os parâmetros da operação. O uso de tipos de dados complexos nas mensagens é descrito no elemento *WSDL:type*. O elemento *WSDL:binding* descreve o protocolo, formato de dados, segurança e outros atributos de uma interface de serviço em particular.

A definição da implementação do serviço é um documento WSDL que descreve como uma particular interface é implementada por algum provedor de serviço (*service provider*). O serviço Web é modelado (Figura 11) como um elemento *WSDL:service* que contém uma coleção (geralmente um) de elementos *WSDL:port*. Uma porta (*port*) associa um *endpoint* (por exemplo, um endereço de rede ou uma URL) com um elemento *WSDL:binding* da definição da interface do serviço.

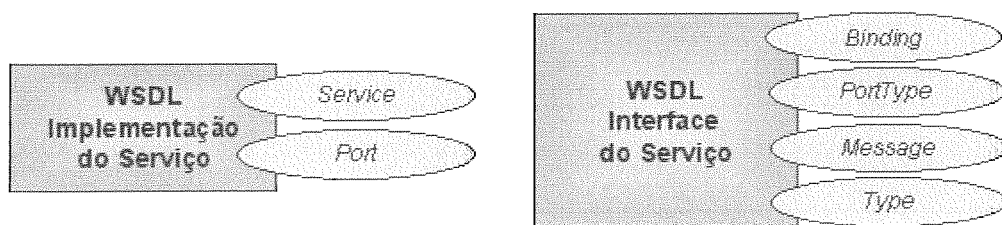


Figura 11 - Implementação e Interface de um serviço em uma descrição WSDL

O SOAP (*Simple Object Access Protocol*) define um protocolo para a troca de informação. O protocolo SOAP estende o XML para que aplicações clientes possam facilmente enviar parâmetros para aplicações servidoras e então receber e entender o documento XML retornado. Graças à simplicidade de um arquivo XML (texto puro), dados são facilmente intercambiados entre sistemas de computadores tendo diferentes arquiteturas e formatos de dados.

Uma vez que um serviço Web é definido usando WSDL, é necessária alguma forma de torná-lo conhecido para utilização. Isso é feito através do registro UDDI (*Universal Description Discovery and Integration*) que fornece métodos para publicar e encontrar descrições de serviços [33]. Dessa forma, provedores de serviços podem publicar a existência de seus serviços para que potenciais usuários os encontrem. O registro UDDI armazena uma descrição do serviço (conforme o tipo de negócio, dividindo por categorias e organizado hierarquicamente) e a localização do mesmo (*binding*).

A arquitetura de serviços Web em conjunto com os protocolos de descrição (WSDL), comunicação (SOAP) e registro (UDDI) habilitam esta tecnologia como uma grande alternativa para ser utilizada no atual ambiente de pesquisa em bioinformática. As descrições WSDL permitem que programas públicos de bioinformática tenham suas interfaces melhor descritas, diminuindo a dificuldade para invocá-los. O protocolo SOAP é apropriado para efetuar a comunicação entre plataformas heterogêneas, que é o caso deste ambiente científico. Por último, o registro UDDI constitui-se em uma ótima alternativa para a catalogação e posterior descoberta dos diversos programas existentes. Além disso, serviços Web também são adequados para a composição de diversos programas em workflows, conforme será visto na próxima seção.

A idéia de utilizar serviços Web ao invés de invocar um programa de bioinformática diretamente é particularmente interessante quando se pensa na interoperabilidade dos diversos programas existentes atualmente neste ambiente. A tecnologia de serviços Web foi especialmente concebida para prover interoperabilidade entre aplicações de diferentes plataformas. Programas podem ser encapsulados como serviços. A utilização desses programas portanto, é feita através de chamadas a serviços e não chamadas diretamente aos programas. Se a localização de um programa muda, apenas a descrição do serviço é modificada e as diferentes chamadas ao serviço ficam

imunes a esta mudança. Serviços Web também são uma solução apropriada para o intercâmbio e reuso de recursos, suportando a colaboração de equipes distribuídas geograficamente. Todas essas vantagens já estão sendo exploradas em muitos projetos de pesquisa que estão começando a utilizar a tecnologia de serviços Web [37][47][89][129][175][179]. Porém, esses projetos não mostraram experimentos de uso real na área de bioinformática, como foi feito nesta dissertação.

3.2 Workflows

Em empresas e organizações existem alguns procedimentos de trabalho que estão bem estabelecidos e normalmente são executados diversas e diversas vezes. Como esses procedimentos estão normalmente relacionados com a forma de funcionamento da empresa eles também são conhecidos como processos de negócio (*business process*) [71]. Um processo de negócio é geralmente composto de um conjunto de atividades que representam uma unidade de trabalho completa e essas atividades podem ser interdependentes uma das outras. O conjunto de atividades envolvidas em um processo de negócio junto com suas entradas e saídas é chamado de workflow [1][82] [103].

Um exemplo típico de workflow pode ser visto no relacionamento entre um comprador e um vendedor (Figura 12). A primeira atividade que inicia o workflow é o pedido de algum produto que o comprador faz ao vendedor (1). Ao receber este pedido o vendedor irá confirmar o pagamento junto a algum serviço de cartão de crédito (2) e após receber a informação de aprovação do crédito (3) irá proceder à confirmação se o produto pedido possui quantidade disponível no estoque (4 e 5). Confirmada também esta etapa, bastará ao vendedor enviar para o comprador a informação sobre a confirmação do pedido e o prazo previsto para entrega (6). Neste exemplo, as atividades de efetuar pedido ao vendedor e confirmar o crédito na instituição financeira estão disponíveis a qualquer interessado e por isso são classificadas como atividades externas. Já a atividade de verificação de estoque só está disponível para o vendedor, pois é um procedimento que, neste caso, só interessa ao mesmo, sendo classificada como uma atividade interna.

O mais interessante disto tudo, é que cada atividade descrita anteriormente poderia estar encapsulada dentro de um serviço Web e dessa forma, se beneficiar da interoperabilidade fornecida por esta tecnologia através da padronização da interface e

da comunicação dos serviços. Dessa forma, os serviços Web podem ser naturalmente utilizados como componentes de software reutilizáveis prontos para serem “*plugados*” em um processo mais abrangente, um workflow, e assim fornecer a total integração entre diferentes tipos de aplicações [126]. O casamento entre serviços Web e workflows é o processo natural de evolução de sistemas distribuídos e atualmente consiste em um grande ramo de pesquisa [158].

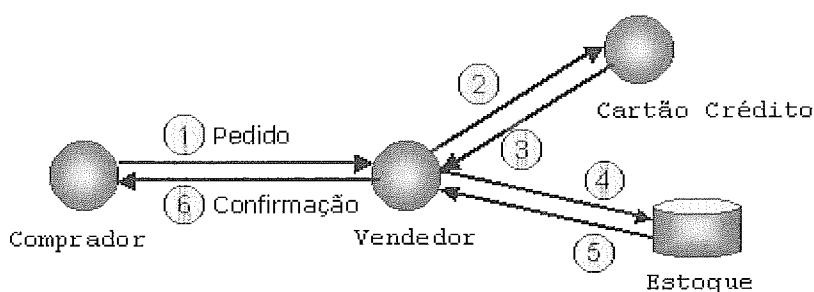


Figura 12 - Exemplo de um workflow em um ambiente de negócio eletrônico (e-business)

O termo workflow é amplamente utilizado para representar vários níveis de conceitos desde o processo de negócio a ser modelado, a especificação deste processo, o software que automatiza a execução deste processo ou até simplesmente o software que suporta a coordenação e colaboração das pessoas que implementam um processo [71]. O mais importante, porém, é distinguir as duas atividades que envolvem o desenvolvimento de um workflow: a especificação da definição do workflow e a execução do workflow propriamente dito [1].

A **especificação de um workflow** é a tarefa de definir o fluxo de execução do mesmo, indicando as atividades envolvidas, as restrições existentes, a ordem de execução, a aquisição, conversão e formatação dos dados de entrada e saída, a existência de possíveis desvios no fluxo normal de execução e possivelmente o tratamento de erros e atividades compensatórias.

Mais formalmente, podemos definir um workflow como uma coleção de atividades organizadas para acompanhar algum experimento científico. As **atividades** ou **tarefas** são os componentes de software independentes que implementam alguma funcionalidade e são executadas por um ou mais sistemas de softwares. Exemplos de atividades incluem executar um programa, transformar um arquivo ou atualizar um banco de dados. Além disso, um workflow define a **ordem de execução** dessas atividades ou as **condições** em que essas atividades serão executadas e a sua eventual

sincronização. Os **dados** de entrada e saída das atividades (variáveis) são definidos como o fluxo de dados do workflow. Resumindo, um workflow W é representado pela quádrupla (T, V, Sf, Cf) onde:

T é um conjunto $\{t1, t2, \dots, tn\}$ de tarefas de W ,

V é um conjunto de variáveis $\{v1, v2, \dots, vn\}$ de W definindo um fluxo de dados,

Sf é uma função sucessora associada a cada tarefa $t \in T$, e

Cf é uma função de condição associada a cada tarefa $t \in T$.

A **execução do workflow** é a execução de todas as atividades definidas na especificação do workflow seguindo a seqüência estabelecida e levando em consideração os quatro elementos definidos anteriormente: é necessário seguir a ordem de execução do processo definido, os dados gerados possivelmente necessitarão de algum tipo de transformação ou conversão para serem trocados entre as tarefas, a invocação poderá ser feita de modo automático ou manual e, por fim, ainda poderá existir alguma ferramenta que monitore a execução e forneça estatísticas sobre ela.

Um sistema gerenciador de workflows (WfMS - *Workflow Management Systems*) é o software que fornece toda a infra-estrutura para definir, executar e monitorar workflows. A utilização deste tipo de software é importante para separar a definição do workflow, do mecanismo (*engine*) encarregado de executar o mesmo, permitindo dessa forma que modificações em uma das partes não afete o funcionamento da outra. Existem diversos sistemas de workflows fornecidos por diferentes empresas como o Lotus Workflow da IBM [88], o MS Message Queuing (MSMQ) da Microsoft [108], o Oracle Workflow da Oracle [116], entre outros. Esses softwares fornecem desde mecanismos para o suporte básico de workflows até soluções completas para a definição, execução e gerenciamento. O mais importante, porém, é o suporte que esses softwares dão a forma de especificar (modelar) o workflow. Essa especificação pode ser feita através de linguagens abertas ou proprietárias, utilizando diagramas, grafos, descrições textuais ou até mesmo documentos XML.

Para facilitar o uso de workflows entre os diversos fornecedores de produtos foi fundada em 1993 uma entidade chamada *Workflow Management Coalition* (WfMC) [153] para tratar da definição de padrões para a utilização e gerenciamento da tecnologia de workflows. O objetivo desta entidade inclui padronizar as especificações de workflows para permitir interoperabilidade entre diferentes fornecedores. Já foram

publicados um glossário WfMC, largamente aceito na indústria como um padrão para a terminologia de workflow, e o modelo de referência WfMC (*WfMC Reference Model*) [82] identificando cinco interfaces funcionais que são relevantes para a integração de sistemas de gerenciamento de workflows com tecnologias complementares ou relacionadas. Outra iniciativa importante é o trabalho de Aalst *et al.* na definição de padrões para workflows (*Workflows Patterns*) [2][178].

Linguagens de Definição de Workflows

Para a especificação da definição de um workflow algum documento deve ser utilizado para descrever o seu fluxo de execução. Diagramas, grafos, documentos XML e outros formatos são utilizados pelos diversos fornecedores de WfMS. À par da diferença entre esses formatos, de uma maneira ou de outra eles precisarão de componentes para a descrição das atividades, do fluxo do processo, das estruturas de controles, dos dados, das transformações entre os dados e adicionalmente, até mesmo de mecanismos de tratamento de erros e de controle de transações.

Diversas linguagens surgiram na literatura para a definição de workflows, mas ainda não existe um consenso sobre uma linguagem padrão. Existem diferentes propostas como ebXML [58], WSFL [104], XLANG [144], WSCL [163] e BPML [13]. Em Bernauer *et al.* [28] existe um ótima comparação entre as diversas abordagens existentes. Recentemente, IBM, Microsoft e BEA lançaram a linguagem BPEL4WS (*Business Process Execution Language for Web Services*) [52]. Também conhecida como BPEL, essa linguagem possui um rico conjunto de construtores XML utilizados para coordenar processos de negócio encapsulados em serviços Web. Como o BPEL é um esforço comum entre grandes fornecedores do mercado e substitui as antigas propostas da IBM (WSFL) e Microsoft (XLANG), ele se torna um sério candidato a se tornar a linguagem padrão para a definição de workflows que utilizam serviços Web.

O padrão OWL-S [118] utilizado para a descrição, descoberta e execução de serviços Web, construída sobre a linguagem de ontologias Web (OWL) [165] também pode ser utilizado para a composição de diversos serviços em um workflow. Na verdade, de acordo com Aalst *et al* [2], independente da quantidade de linguagens existentes, a maioria delas suporta os padrões principais de modelagem de workflows, o

que poderia dar origem a modelos canônicos para o qual essas linguagens poderiam ser convertidas e mapeadas.

Workflows Científicos

Diversas classificações para workflows existem na literatura [71] [110]. Sistemas gerenciadores de workflows autônomos são sistemas totalmente funcionais sem a necessidade de outras aplicações a não ser o servidor de banco de dados e a camada de troca de mensagens. Esses sistemas ficam responsáveis por invocar as aplicações externas que participam de um workflow, passando os dados relevantes entre os envolvidos. Eles normalmente possuem suas próprias interfaces com o usuário e suportam diferentes tipos de aplicações. Sistemas gerenciadores de workflows embutidos, por outro lado, como o nome já diz, estão associados diretamente a uma aplicação, como por exemplo um sistema ERP (*Enterprise Resource Planning*). As funcionalidades do workflow estão espalhadas por esses tipos de sistemas fornecendo suporte para controlar a seqüência de aplicações, gerenciar as mensagens e processar as exceções.

Outra classificação muito importante e que está **diretamente relacionada com este trabalho** se refere aos workflows científicos. Workflows científicos compartilham diversas características com workflows tradicionais utilizados na área comercial (*business workflows*), mas possuem outras necessidades [135]. O termo workflow científico é utilizado para descrever o processo de workflow voltado para as aplicações científicas de áreas de pesquisa como biologia, física, astronomia, geologia entre outras. Essas aplicações são baseadas em experimentos científicos com alto poder computacional e que normalmente são utilizados para criar e validar alguma nova descoberta na forma de novos modelos científicos, que representam uma abstração de maior nível para um domínio de conhecimento específico.

Por tratar da busca de novos conhecimentos, um mesmo experimento científico normalmente é executado inúmeras vezes com apenas algumas variações, em um processo interativo e incremental. Dessa maneira, pode-se dizer que a definição do workflow muda constantemente. Além disso, tanto as repostas positivas do workflows, como as respostas negativas são importantes na análise do cientista. Também é de

extrema importância, manter o registro completo de como cada resultado foi obtido, pois pode ser necessário reproduzir o experimento.

Workflows científicos precisam um grande suporte ao processo de especificação, já que a busca da melhor seqüência de execução nem sempre é clara. Para o caso de cientistas e biólogos, a definição de um workflow envolve a tomada de diversas decisões, análises e trabalho de equipe, bem diferente de um workflow comercial que foi modelado para atender a um processo de negócio relativamente fixo, como processar um pedido, verificar o estoque, enviar a mercadoria e emitir a fatura. Na execução, um mesmo biólogo executa várias vezes no mesmo dia o mesmo workflow mudando apenas alguns parâmetros. Workflows comerciais são executados várias vezes, mas normalmente invocados por usuários diferentes, como por exemplo, os vários clientes que fazem um pedido em uma loja.

Para tratar todas essas características, os workflows científicos deverão possuir diversas funcionalidades a mais em relação aos workflows da área comercial, como visto em [171] e comentados a seguir.

- *Execução Parcial*: como a definição do workflow é um processo de aprendizado, os cientistas só poderão decidir como continuar um workflow após terem avaliado os resultados parciais.
- *Modificações Dinâmicas*: a definição do workflow é um processo dinâmico que é influenciado pelos resultados obtidos, gerando constantes mudanças no fluxo de execução.
- *Reutilização*: workflows já executados poderão ser reutilizados para reproduzir um antigo experimento. Dessa forma, é necessário armazenar as entradas, parâmetros e resultados finais e parciais de cada execução.
- *Aprendizado com os erros*: o processo de definição de workflows é baseado na tentativa e erro fazendo com que a análise e armazenamento de resultados negativos sejam tão importantes quanto os resultados positivos.
- *Documentando como foi feito*: manter informações sobre de onde vieram os dados, onde e quais as transformações que foram feitas neste dado e como cada resultado foi obtido é de extrema importância para a análise final.

Por fim, podemos generalizar o ciclo de vida de um workflow científico em cinco fases:

Fase 1: Modelagem do workflow. O workflow é definido usando alguma representação gráfica ou linguagem como uma coleção de atividades agrupadas, utilizando programas ou serviços pré-existentes.

Fase 2: Publicação do workflow. O workflow é publicado também como um serviço, ou seja, fica disponibilizado para ser encontrado e utilizado, até mesmo por outros workflows.

Fase 3: Execução do workflow. Execução do workflow através da execução de cada uma de suas atividades, onde a saída de uma atividade, normalmente, corresponde a entrada da próxima. Testes no fluxo de execução, desvios, restrições, tratamentos de erros e outras características são comuns na execução de um workflow.

Fase 4: Visualização dos resultados. Os resultados finais e parciais podem ser visualizados e analisados podendo gerar ajustes e novas re-execuções do workflow.

Fase 5: Finalização do workflow. Podemos considerar como o último passo onde o experimento modelado pelo workflow atingiu o seu objetivo, produziu resultados relevantes e algum tipo de conhecimento científico poderá ser inferido do mesmo.

Também é importante diferenciar os tipos de usuários que podem estar envolvidos nos experimentos *in silico*. Existe uma separação bem clara do usuário que define o workflow, isto é, monta a seqüência de etapas que o workflow possuirá, e o usuário que executa o workflow propriamente dito. Apesar de em alguns casos a mesma pessoa representar ambos os papéis, fica claro visualizar a diferença entre eles, ao atribuir as responsabilidades de cada um. O usuário que define o workflow é responsável pelas fases 1, 2 e 5 descritas anteriormente, enquanto que o usuário que executa é responsável pelas fases 3 e 4.

3.3 10+C: As 10 Principais Características Para Ambientes de Experimentos *In Silico*

No contexto dos experimentos científicos de bioinformática, poderíamos agrupar as características necessárias para este tipo de ambiente em três categorias: definição, execução e ambiente de utilização. Na categoria definição ficariam as características que promovem facilidades para a descrição de workflows pelos cientistas. A categoria execução concentra todas as facilidades quanto à execução e registro de experimentos e

por último, a categoria do ambiente de utilização diz respeito às facilidades que os usuários irão ter para utilizar o ambiente. Com essas três categorias em mente, definimos dez características significantes que qualquer solução para ambientes de experimentos *in silico* deve possuir. São elas:

- (i). Definição abstrata do workflow;
- (ii). Definição do workflow em nível de programas executáveis;
- (iii). Geração automática do workflow;
- (iv). Execução do workflow;
- (v). Re-execuções totais e parciais do workflow;
- (vi). Tratamento de exceções na execução do workflow;
- (vii). Registro da execução de programas;
- (viii). Registro da execução de workflows;
- (ix). Execução remota de programas utilizando padrões abertos;
- (x). Distribuição através da Internet;

Definição abstrata do workflow: É necessária alguma abstração de mais alto nível que permita isolar a definição do workflow, da invocação de cada um dos programas. Quando se utilizam linguagens de *script*, tanto a definição, como a execução, pertencem a um mesmo código fonte. Esta abordagem torna o fluxo do workflow de difícil entendimento e modificação. A elevação da especificação de uma seqüência de atividades para uma representação de maior nível de abstração, permite aos cientistas maior clareza na visualização do processo como um todo, na elaboração de processos alternativos e na detecção de possíveis falhas no processo atual. A utilização de um documento específico para a especificação do workflow permite na maioria das vezes a utilização de ferramentas gráficas para a definição do workflow, através de diagramas, polígonos, setas, etc, para representar cada uma das atividades e o fluxo de execução.

Definição do workflow em nível de programas executáveis: a definição de alto nível, discutida no item anterior, precisa, de alguma forma, casar com os programas que serão realmente executados, ou seja, os programas que serão fisicamente invocados e que produzirão algum resultado que será composto no workflow para a produção do resultado final. Esses programas são, por exemplo, o programa BLAST do instituto NCBI de uma máquina específica, ou o programa WU-BLAST de outra máquina. Assim, a definição do workflow deve ser feita diretamente no nível de programas

executáveis ou de alguma maneira, mapeando a definição abstrata a este nível mais baixo, para desta forma, permitir a execução do workflow pelo mecanismo de execução responsável.

Geração automática do workflow: a definição do workflow é o ato de representar o fluxo de execução de uma seqüência de atividades. Essa representação pode ser feita utilizando linguagens de alto nível ou até mesmo algumas ferramentas gráficas de representação. De uma forma ou de outra, pode ser necessário que esta representação seja importada para dentro do mecanismo de execução do workflow ficando disponível para o uso posterior. Dessa forma é essencial que não apenas exista a representação em nível mais alto do workflow, mas que esta representação de alguma maneira seja automaticamente gerada, publicada e disponibilizada para a efetiva utilização do workflow.

Execução do workflow: A execução do workflow será efetuada a partir da sua definição em nível de programas executáveis discutido anteriormente. Através de um mecanismo específico responsável pela execução do workflow, o conjunto de aplicações será executado seguindo a seqüência de atividades previamente definidas. Dados e parâmetros de entrada deverão ser escolhidos para serem utilizados durante a execução.

Re-execuções totais e parciais do workflow: característica essencial na execução de experimentos científicos, pois é ela que permite o ciclo de “experimentação e teste” inerentes ao processo de descoberta do conhecimento. Re-execuções de instâncias do workflow com diferentes parâmetros ou diferentes entradas de dados, desde o início do workflow, ou de um ponto qualquer até outro, permite aos cientistas um grande auxílio e aumento de produtividade no seu dia a dia de trabalho.

Tratamento de exceções na execução do workflow: Essa é uma característica essencial para o bom andamento dos experimentos *in silico*, já que permite a definição de atividades compensatórias caso algum erro aconteça no fluxo normal de execução. Dessa forma, programas alternativos podem ser executados caso o programa principal apresente algum erro ou não esteja disponível;

Registro da execução de programas: o armazenamento dos dados referentes às execuções dos programas individuais, desde parâmetros e dados de entrada, até os

resultados obtidos, representa uma forma eficaz de manter-se um registro para futuras consultas e análises de cada aplicação.

Registro da execução de workflows: o armazenamento de todo o conjunto de programas que fazem parte de um workflow representa um aumento de confiabilidade nos experimentos, além de abrir novas possibilidades de pesquisa, pois os dados armazenados podem ser analisados através de técnicas de mineração de dados para a busca de novos conhecimentos. O armazenamento desses dados é o que permite a possibilidade de re-execuções totais e parciais de uma instância de workflow já executada. Também é de grande interesse armazenar os metadados referentes as execuções do workflow, como por exemplo quem executou, quando, como, quais programas foram utilizados em cada instância do workflow e a partir de onde foram invocados, entre outros, permitindo analisar a proveniência dos dados.

Execução remota de programas utilizando padrões abertos: é fundamental para o sucesso de qualquer solução a utilização da base de programas já existentes na comunidade científica. Para isso, esses programas devem estar publicados e acessados através de algum tipo de padrão aberto e, preferencialmente, amplamente utilizado. Neste contexto, as soluções devem ser capazes de entender esses padrões para acessar e invocar cada um dos programas. Atualmente, o padrão de serviços Web é o que mais se aproxima de uma solução tecnológica flexível e que fornece alto grau de interoperabilidade entre aplicações.

Distribuição através da Internet: o ambiente onde será feita a gerência do workflow é uma questão essencial e pode ser considerada quase como uma resposta unânime: a utilização intensiva da Internet permite que qualquer solução possa ser compartilhada automaticamente por toda a comunidade científica ao redor do mundo. Neste ponto, tanto a arquitetura de serviços Web (WSA, *Web Service Architecture*) [160] como a arquitetura de serviços em Grids Computacionais (OGSA, *The Open Grid Service Architecture*) [117] representam a base tecnológica que devem ser utilizadas para as soluções propostas.

A computação em Grid pode ser definida como a infra-estrutura de computação distribuída que permite o compartilhamento coordenado, flexível e seguro de recursos para uma comunidade dinâmica de indivíduos e instituições [65]. Tem o foco em

aplicações e recursos de alto poder computacional orientados para a resolução de problemas complexos e em larga escala.

3.4 Trabalhos Relacionados

O problema de integração de bases de dados e interoperabilidade entre aplicações não é novo e muito menos exclusivo da área de bioinformática. Há vários anos questões envolvendo banco de dados heterogêneos e distribuídos vêm sendo tratadas na literatura [119]. A abordagem que utiliza mediadores [173] como forma de controlar o acesso ao banco de dados e distribuir as consultas pelas bases remotas através da utilização de tradutores (*wrappers*) é útil na obtenção de resultados envolvendo várias fontes de dados mas possui grande dificuldades em tratar o problema de evoluções no esquema dessas bases.

A interoperabilidade entre aplicações ou componentes de software também já foi tratada sob diversas óticas e tecnologias diferentes, desde soluções de código proprietário como CORBA [114] e DCOM [107], até soluções de código aberto específicas para uma determinada plataforma, como JAVA e RMI [140]. Recentemente, com o advento da Internet e seus padrões abertos da W3C, surgiu a abordagem de serviços Web, que pretende fornecer uma padronização para a interoperabilidade entre aplicações utilizando a Internet e protocolos como XML, SOAP e WSDL, como visto na seção 3.1.

Entretanto, como visto na seção 2.3, um dos maiores desafios da comunidade de bioinformática se refere à composição de diversas aplicações e banco de dados em um fluxo contínuo de execução, onde a saída de uma aplicação pode ser usada como entrada para a próxima, através de workflows científicos para a definição e execução de experimentos *in silico*.

Existem diversas soluções na literatura para Sistemas de Gerenciamento de Workflows (WfMS). Entretanto muitos deles são genéricos para qualquer tipo de aplicação [88], alguns possuem alto custo comercial [108][116] e muitos não são específicos para a área de bioinformática [22] e por conseqüente não são ideais para este tipo de ambiente, como vimos na seção 3.2.

LabBase/LabFlow

Este projeto pode ser considerado uma das primeiras iniciativas que tratou o problema dos experimentos científicos. Ele possui uma divisão clara entre o banco de dados que armazena os dados sob um esquema específico que descreve um experimento, chamado LabBase [73] e o mecanismo responsável pela execução do workflow, chamado LabFlow [74].

O LabBase provê um modelo de dados de objetos estruturados que serão utilizados na descrição dos objetos do banco de dados através dos conceitos de *Materiais (Materials)*, *Passos (Steps)* e *Estados (States)*. *Materiais* são objetos que representam itens tangíveis que participam dos procedimentos em um laboratório, como clones ou seqüências. *Passos* são objetos associados ao resultado de um experimento como o seqüenciamento de um clone ou a execução do programa BLAST em uma seqüência. Por fim, o objeto *Estado* representa a situação que os *Materiais* podem assumir durante um experimento, como por exemplo “pronto para seqüenciamento” ou “pronto para análise do BLAST”. Todos esses conceitos possuem relacionamentos entre si no banco de dados, dessa forma, um *Passo* sempre possuirá o *Material* em que foi executado e o *Material* estará relacionado com os *Estados* anterior e posterior a essa execução.

O LabBase é implementado em Perl como uma camada por cima de um sistema gerenciador de banco de dados relacional. Foi inicialmente desenvolvido para Sybase e então portado para Oracle.

Os programas e dados utilizados devem estar descritos através dos três conceitos básicos da arquitetura, o que representa uma desvantagem para a utilização de aplicações existentes. O experimento pode ser descrito em alto nível através dos vários *Passos* que um *Material* poderá possuir. Entretanto, esta definição está presa aos programas específicos que foram definidos nos passos.

O experimento é executado através do sistema gerenciador de workflows LabFlow. Este sistema está interligado com o banco LabBase para utilizá-lo como forma de suportar todo o armazenamento dos objetos e do histórico da execução. A execução de um workflow está ligada a idéia que todo *Material* está em um determinado *Estado* e quando o *Material* é processado, ele se move de um *Estado* para

outro. O estado do Material determina em que fase do workflow o mesmo se encontra e qual será o próximo *Passo* a ser aplicado. Por exemplo, os possíveis *Estados* para uma cadeia de um DNA poderiam incluir: *esperando_por_sequenciamento*, *sequenciamento_inicial*, *sequenciamento_final*, *sequenciamento_completado*. Dessa forma, um workflow poderia ser definido como um grafo que leva o *Material* do *Estado esperando_por_sequenciamento* até o *Estado sequenciamento_completado*, passando por *sequenciamento_inicial* e *sequenciamento_final*.

O projeto Labbase/LabFlow tem grandes méritos por ser um dos pioneiros na adoção de workflows para experimentos científicos e inspirou diversos outros projetos de pesquisa. Entretanto, por ser baseado na linguagem Perl e utilizar sistemas fechados como Oracle, prejudicam a sua flexibilidade e adoção sistemas reais de bioinformática. Para a definição do workflow, apesar de defini-los utilizando os conceitos abstratos de *Passos* e *Materiais*, não existe nenhum mecanismo que auxilie o biólogo nesta tarefa. A execução do workflow também não é totalmente automatizada, existindo a necessidade de consultar quais *Materiais* estão em determinado *Estado* (inicial, por exemplo) e qual o *Passo* que pode ser executado para este Material neste *Estado*. Também não existe suporte a re-execuções totais e parciais e nem ao tratamento de erros. Um ponto positivo deste projeto é o armazenamento dos resultados referentes às execuções dos programas e dos workflow através dos mesmos objetos *Materiais* e *Estados*, possibilitando futuras consultas e o rastreamento de como o experimento foi feito, o que é fundamental para a identificação da procedência desses dados.

Chimera

O sistema Chimera [66], pertencente ao projeto GriPhyN [76], tem o objetivo de ser um repositório para representar, consultar e executar tanto os dados envolvidos em procedimentos computacionais científicos, como os próprios procedimentos computacionais. A representação e o armazenamento dessas informações pode permitir a possibilidade do rastreamento da origem dos dados (*data provenance*) [34][35] e habilitar novas descobertas.

Os dados são organizados no chamado catálogo de dados virtual (*Virtual Data Catalog* ou VDC) na forma de três entidades: *Transformações*, *Derivações* e *Objetos de Dados*. *Transformações* representam um programa que deverá ser executado. Ele é

descrito através de algumas características específicas (nome do autor, tipo, versão) junto com as informações necessárias para o mesmo ser invocado (nome do executável, argumentos, ambiente). *Derivações* representam uma execução de uma *Transformação* junto com todos os valores específicos utilizados nessa transformação (valores de argumentos, tempo de execução), ou seja, uma instância da *Transformação*. Os *Objetos de Dados* são as entidades que serão consumidas ou produzidas por uma determinada *Derivação*. Além disso, uma *Transformação* pode ser estendida para uma *Transformação Composta* que serve para descrever a execução de múltiplos programas.

Além do catálogo de dados virtual (VDC) o sistema também provê uma linguagem para permitir consultas e requisições ao repositório. Essa linguagem é chamada de linguagem de dados virtual (*Virtual Data Language*) ou VDL e é baseada no SQL. O sistema Chimera também pode ser disponibilizado como um componente da arquitetura de Grid de Dados [64], ser distribuído através de diferentes localizações e ser integrado a outros recursos de um sistema de Grid.

O sistema Chimera, portanto, é distribuído através de uma tecnologia de código aberto e não proprietária através de Grids Computacionais, mas o acesso é feito somente através da VDL. Para a definição dos workflows, os conceitos de *Transformações* simples e compostas podem ser utilizados para uma especificação de alto nível e a existência da diferença entre os conceitos de *Transformações* e *Derivações* permite claramente a definição abstrata do workflow, ou seja, sem estar presa a um programa específico. Esta definição pode ser representada através de grafos de execução abstratos que então são transformados em códigos executáveis e submetidos ao Grid via Condor-G [49] e o sistema de escalonamento DAGman [53] para a execução do workflow.

Entretanto não foram encontradas descrições de mecanismos que suportem re-execuções parciais de trechos do workflow. Ele também só possui registro de execuções em nível de programas, não possuindo a mesma facilidade para os workflows, minimizando a possibilidade da identificação da proveniência dos dados. Para o armazenamento dos dados originados através dos *Objetos de Dados*, a aplicação cliente deve ser responsável em após a execução de cada passo do workflow, armazenar os dados intermediários no catálogo do Chimera.

MyGrid

MyGrid [179] é um projeto de pesquisa para prover um ambiente de Grid Computacional de alto nível para aplicações de bioinformática, com foco na integração de dados, workflows e procedência dos dados. O myGrid utiliza ontologias [77][78] para descrever, descobrir e compor serviços em um ambiente de experimentos científicos.

Ontologias provêem um vocabulário de termos e conceitos para formar descrições. A utilização de textos livres para descrever algum conceito é poderosa e flexível, porém encontra grande dificuldade para efetuar buscas e classificações. Uma melhor alternativa é a utilização de um núcleo de vocabulário junto com a gramática e as regras para formar novas palavras, permitindo assim a organização dos termos em uma estrutura hierárquica de classificação. MyGrid utiliza um conjunto de ontologias para representar os metadados, que como visto em [40], possui grande importância para o gerenciamento de recursos científicos.

A utilização de ontologias é importante para adicionar mais semântica na descrição e classificação dos recursos científicos e dessa forma tornar mais fácil o processo de descoberta de qual recurso utilizar em um determinado experimento ou escolher qual aplicação executar dentre diversas aplicações similares. Para descrever os serviços é utilizada a linguagem de ontologia DAML+OIL [83] baseada na linguagem DAML-S [10] e agrupadas em duas categorias: metadados de domínio e metadados de negócio. Metadados de domínio estão relacionados à semântica que o serviço possui como por exemplo, àquela que descreve a aplicação BLASTn como uma ferramenta para calcular a homologia de seqüências que usa o algoritmo BLAST executado sobre uma seqüência de nucleotídeos. Metadados de negócio estão relacionados ao acesso ao serviço como localização geográfica, custo, qualidade, etc.

O MyGrid possui uma seqüência de ontologias para representar os diversos recursos (programas, dados, workflows) dos experimentos em bioinformática. Na verdade, os metadados estão organizados em quatro níveis hierárquicos: classe de serviços, serviço abstrato, serviço instanciado e serviço invocado. Classes de serviços descrevem o serviço genericamente em algum domínio (ex: banco de dados de proteínas). Um serviço abstrato especializa alguma classe de serviço em um serviço bem

definido (definido parâmetros) mais ainda abstrato (ex: o banco de dados de proteínas SWISS-PROT). Um serviço instanciado já concretiza algum serviço abstrato como o banco de dados SWISS-PROT da instituição EBI (*European Bioinformatics Institute*). Um serviço invocado descreve uma execução de um serviço instanciado, junto com os valores dos parâmetros, uma data em particular, etc. Essa hierarquia é utilizada para achar serviços alternativos antes ou durante a execução do workflow.

Além dos serviços, os workflows também podem ser descritos através da ontologia DAML-S. A utilização dos conceitos de operações atômicas e operações compostas (que são uma cadeia interligada de operações atômicas) permitem a ordenação da execução do processo. Entretanto, não existe suporte na linguagem DAML+OIL para definir processos de controle desses workflows e também não existe mecanismos capazes de interpretar e executar essas descrições para o ambiente Grid. Dessa forma, o projeto myGrid utiliza a linguagem de definição de workflow WSFL para representar os controles e fluxos de execução, deixando apenas uma mínima representação em DAML-S.

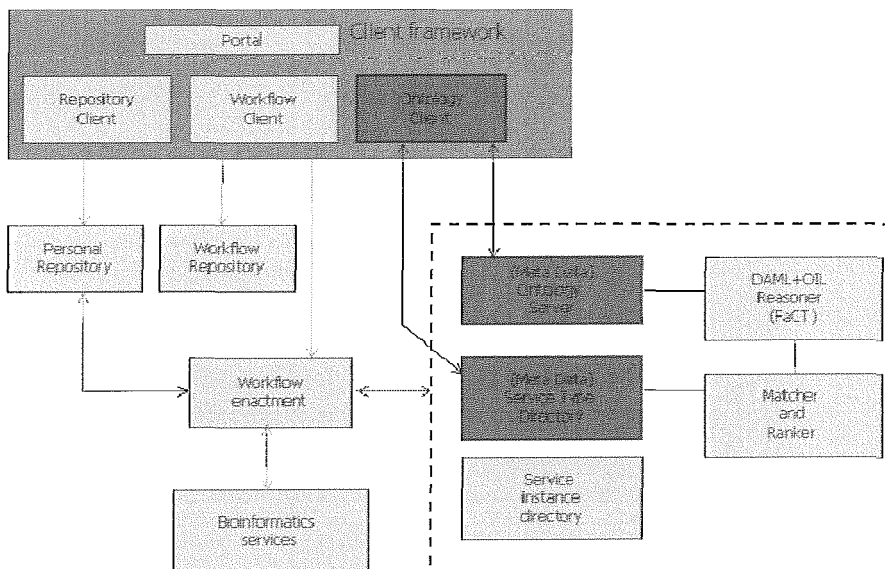


Figura 13 - Arquitetura de funcionamento do myGrid [179]

A arquitetura do myGrid é composta de quatro componentes principais: portal web que provê a interface com os usuários; o repositório do usuário que armazena as informações pessoais; o repositório de workflows que armazena os workflows; e o mecanismo de execução do workflow responsável por interpretar as definições e invocar os serviços específicos no Grid, como mostrado na Figura 13. Além desses, existe uma

série de componentes que são responsáveis pela descrição e descoberta dos serviços, representados pelo quadrado em linha pontilhada na Figura 13.

O projeto myGrid é um dos mais completos na literatura atual. Ele possui funcionalidades para a definição em alto nível e abstrata de workflows e execução dinâmica dos mesmos. Utiliza serviços publicados como serviços Web e utiliza a infraestrutura de Grids computacionais. Porém, não foram encontradas descrições que mostravam como é feita a geração do arquivo WSFL de definição do workflow, sendo a princípio, o usuário responsável por isso. Além disso, este trabalho carece de experimentos reais na área de bioinformática evidenciando a efetividade da aplicação dessas tecnologias para mostrar a viabilidade deste projeto em um ambiente de produção.

ARSENAL

ARSENAL [96] (*A Repository Supported Earth Observation Application*) é uma proposta da Universidade de Stuttgart para auxiliar a Agência Espacial Européia (ESA) nos projetos de observação e análise de imagens geográficas geradas pelos satélites que orbitam a Terra. A captura, análise, armazenamento e a disponibilização de imagens de satélite para usuários e centros de pesquisa envolvem diversas tarefas e configura-se em um processo extremamente trabalhoso e complexo. O projeto ARSENAL é composto por um *framework* que tem como objetivo integrar essas diversas aplicações através da modelagem de fluxos de processos ou simplesmente, workflows. Embora não voltado à bioinformática, este projeto se relaciona às tecnologias estudadas e apresenta soluções possíveis de generalização para workflows científicos.

As aplicações de observação da Terra (*Earth Observation* - EO) são publicadas através de serviços Web que na verdade representam um workflow inteiro. Essas aplicações uma vez invocadas acessam o repositório EO (Figura 14) a fim de obter o arquivo de definição do workflow descrito na linguagem WSFL. Então, este arquivo é passado para o mecanismo de execução do workflow (Figura 14) que fica encarregado de invocar cada serviço Web em particular e retornar o resultado final. As aplicações EO podem ser acessadas através de um servidor HTTP ARSENAL que é responsável em fazer a interface com o usuário final (Figura 14).

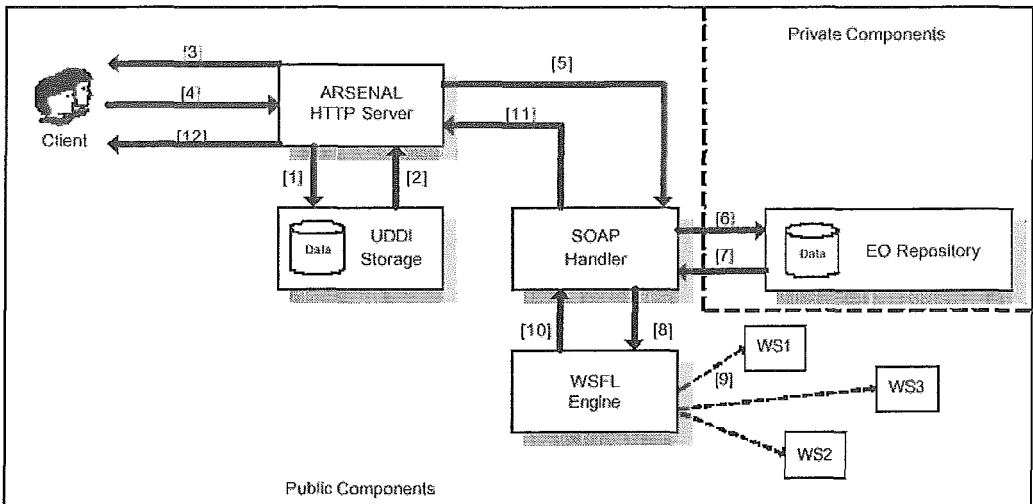


Figura 14 - Arquitetura de funcionamento do ARSENAL [96]

Essa solução utiliza serviços Web, tanto para acessar as aplicações que pertencem ao workflow, como para disponibilizar o próprio workflow, da mesma forma que foi feito neste trabalho. O tratamento de exceções na execução do workflow é previsto na linguagem WSFL habilitando esta funcionalidade neste projeto. Entretanto não há mecanismo para a geração automática dos arquivos WSFL e nem a possibilidade da escolha dinâmica do workflow a partir de uma descrição abstrata e também não existe a preocupação em armazenar no repositório os dados intermediários, os resultados finais e os metadados referentes a cada execução dos programas e do workflow. O repositório armazena apenas a descrição das aplicações EO que estão disponibilizadas na arquitetura. Por conseqüência, não efetuando o registro desses dados não há a possibilidade de efetuar re-execuções totais e parciais do workflow.

Por fim, apesar de o projeto ARSENAL estar mais voltado para a área de astronomia, ele poderia ser ampliado para atender outras áreas, como a bioinformática.

gRNA

Outro trabalho na área de bioinformática é a arquitetura gRNA [98] (*The Genomics Research Network Architecture*). O ambiente gRNA está basicamente dividido em dois componentes: um ambiente para integrar diversas fontes de dados de bioinformática heterogeneamente distribuídas e um ambiente de utilização para modelar, consultar e integrar recursos distribuídos. O ambiente de utilização inclui uma API específica para workflows que serve para a composição de diferentes fontes de

dados biológicas com as várias aplicações de bioinformática existentes, fornecido pelo sistema HyperThesis [30].

A arquitetura do gRNA opera um ambiente de computação de *clusters* constituído de múltiplos computadores chamado de gRNA Grid. A comunicação entre aplicações clientes com o *cluster* acontece através de *Enterprise Java Beans* (EJBs) [90] hospedados em um servidor de aplicação. Existe um linguagem de consultas XML, chamada de XomatiQ [29] para lidar com consultas sobre as diversas fontes de dados distribuídas gerenciadas pelo gRNA.

O sistema HyperThesis provê um ambiente gráfico, totalmente escrito em Java, que permite a criação de workflows através de um interface visual de arrastar-e-soltar, fazendo a composição de atividades, consultas e fontes de dados. Cada atividade é definida com um componente do workflow que podem ser interconectadas para executar uma tarefa. Existem dois tipos de componentes de workflow no HyperThesis: os componentes envolvidos na instanciação de novos objetos e os componentes que executam métodos de outros componentes que foram passados como entrada. O mecanismo de execução de HyperThesis é responsável por executar o conjunto de componentes que formam um workflow. Cada componente possui uma situação que indica se o mesmo pode ou não ser executado. Essa situação é indicada quando todas as entradas de dados de um componente estão disponíveis.

O gRNA, através do HyperThesis, provê um ambiente robusto para a definição de workflows e a sua posterior execução. Entretanto, possui algumas deficiências. A utilização deste ambiente não adota padrões abertos, obrigando a criação dos componentes de workflows fisicamente dentro do ambiente HyperThesis, utilizando a linguagem Jython e usando as bibliotecas já existentes neste ambiente. Essa característica limita a utilização dos programas existentes escritos em outras linguagens com o código fonte disponível ou não, já que os mesmos precisariam ser reescritos em Jython para serem utilizados. Além disso, o ambiente não provê as facilidades de registrar a execução de programas e workflows no mecanismo de execução, o que é essencial para a análise de dados pelos biólogos. Também não existe nenhum mecanismo de tratamento de erros na execução do workflow.

BioMoby

O projeto BioMoby [175][176] é uma proposta para a utilização de serviços Web e XML para a representação e integração de dados e serviços de bioinformática. O objetivo do projeto BioMoby é prover uma arquitetura onde os dados biológicos existentes possam ser representados em um formato comum, adicionando semântica aos mesmos para facilitar o processo de busca e descoberta.

A arquitetura do BioMoby é dividida em três componentes principais: Objetos e Serviços Moby, o Repositório Central Moby e a Hierarquia entre Objetos e Serviços. Os objetos e serviços são utilizados para descrever serviços Web existentes através de documentos XML utilizando uma ontologia pré-existente. Esta descrição por meio de ontologias é que permite dar maior significado aos objetos e serviços registrados na arquitetura. Eles são registrados no repositório central Moby que provê o acesso centralizado e a possibilidade de descobertas de serviços. Por último, toda a descrição semântica desses objetos e serviços deve ser feita seguindo um relacionamento hierárquico entre eles.

Além de descrever serviços Web existentes, a central Moby, assim como um registro UDDI, também é acessado via mensagens SOAP para prover a busca e descoberta de serviços Web.

O projeto BioMoby possui uma grande infra-estrutura no repositório central Moby com bastante documentação e exemplos de código. O foco principal deste projeto é a descrição com maior semântica de serviços de bioinformática pré-estabelecidos através de ontologias e neste aspecto não há como negar a relevância deste trabalho. Entretanto, não existe neste projeto, nenhuma funcionalidade para a definição, execução e registro de workflows.

SRMW

Um trabalho pioneiro e que motivou diretamente o desenvolvimento desta dissertação de Mestrado, é a arquitetura SRMW (*Web Services based Scientific Resources Management*) [43] originada da tese de Doutorado de Cavalcanti [40]. A SRMW é uma proposta para apoiar a realização de experimentos através da gerência de

recursos científicos, de forma a facilitar o intercâmbio, reuso e disseminação de dados, programas, workflows e modelos, fortemente apoiada no uso de metadados.

O gerenciamento destes recursos é vital para o bom andamento de um laboratório *in vitro* de pesquisa científica. Considerando que programas e dados são os seus recursos mais valiosos, devem existir mecanismos que permitam a execução de tarefas do tipo:

- Identificar o programa certo para um determinado experimento;
- Lidar com múltiplas versões de dados e programas;
- Consultar a documentação de experimentos anteriores;
- Encontrar modelos científicos, suas aplicabilidades e onde já foram utilizados com sucesso;
- Saber como (em que ordem) executar uma seqüência de programas, e enfim;
- Permitir a colaboração entre cientistas através da troca destes recursos entre si.

A SRMW caminha nesta direção, através da utilização de descrições semânticas de mais alto nível para os recursos científicos. Cada recurso científico é categorizado na arquitetura segundo o metamodelo SPMW (*Web Services based Scientific Publishing Metamodel*) [42]. Neste metamodelo, diferentes níveis de abstração estão claramente explicitados para descrever programas e dados (Figura 15). No nível operacional, um programa pode ser visto como um código executável gerado pela compilação de um arquivo fonte e os dados seriam os arquivos propriamente ditos. O nível descritivo serviria para representar estes mesmos programas e dados como uma descrição da interface dos programas ou a descrição da estrutura do dado. Em um nível mais alto, teríamos os recursos de programas e recursos de dados que serviriam para agrupar mais genericamente recursos semelhantes.

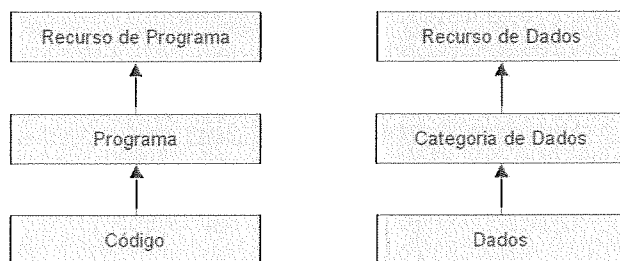


Figura 15 - Níveis de abstrações de programas e dados no metamodelo SPMW

Com estes três níveis de hierarquia, poderíamos, por exemplo, descrever o BLAST no nível de recurso de programa como um algoritmo para alinhamento de

seqüências (o programa FASTA também teria esta mesma classificação). No nível de programa teríamos a descrição do BLAST propriamente dito, com a descrição de suas entradas e saídas e no nível de código, teríamos um código BLAST que está rodando em uma máquina específica.

A arquitetura SRMW possui três componentes na camada de interface com o usuário: os módulos de Publicação, Navegação e Experimentação, acoplados à arquitetura de serviços Web. (Figura 16). O módulo de Publicação é usado para prover recursos descritivos sobre aplicações, dados e workflows disponíveis através de serviços Web. O módulo de Navegação é usado para percorrer os diferentes recursos científicos e suas respectivas descrições a fim de encontrar o mais adequado. Por fim, o módulo de Experimentação interage com o usuário, ajudando-o no processo de instanciação de experimentos executando o mapeamento do nível mais descritivo para o nível operacional de fato.

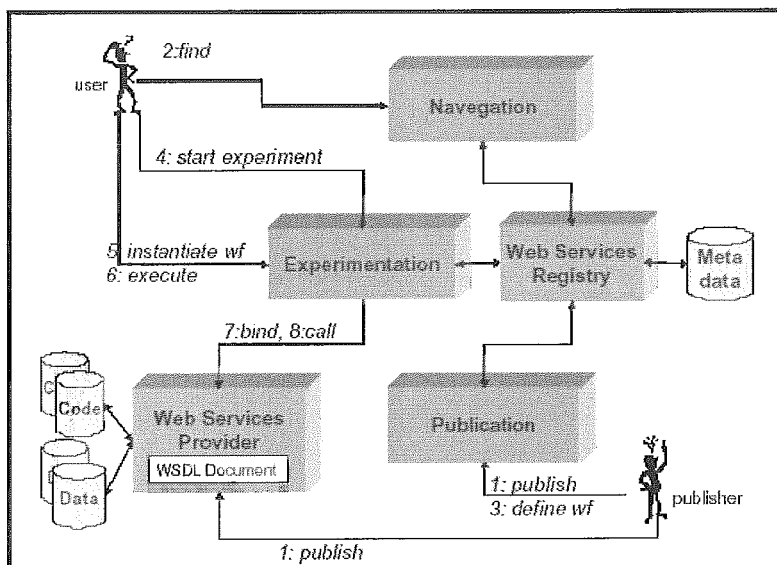


Figura 16 - Módulos da arquitetura SRMW [40]

Através desta arquitetura, a SRMW consegue fornecer o total apoio aos cientistas na criação e execução de experimentos científicos. Sem um ambiente deste tipo, mesmo o especialista mais experiente, encontraria dificuldades para escolher o melhor dado ou o programa mais adequado ao seu experimento. Além disso, gerenciar os resultados de várias execuções de um experimento e utilizá-los para o refino do mesmo, pode ser uma tarefa extremamente trabalhosa, mesmo para o cientista mais organizado. Com a SRMW, essas dificuldades seriam evitadas deixando o cientista livre para o seu trabalho

de pesquisa. Além disso, a infra-estrutura de metamodelos SPMW serve também para a descrição dos próprios experimentos e dos modelos científicos pesquisados.

Entretanto, por se concentrar no nível da gerência dos metadados destes diversos recursos científicos, a SRMW não provê o suporte no nível operacional para a descrição e execução de experimentos através de workflows científicos. Neste contexto podemos afirmar que o Ambiente 10+C e a arquitetura SRMW se complementam, como visto em Cavalcanti, Targino *et al.* [44], para fornecer um suporte efetivo aos laboratórios científicos virtuais.

Discussão sobre os Trabalhos Relacionados

Os trabalhos relacionados foram analisados com base nas dez características apresentadas no início desta seção. A Tabela 1 resume esses trabalhos indicando se os mesmos atendem (✓) ou não (×) essas características.

Os projetos LabBase/LabFlow, Chimera, Arsenal e gRNA se concentram na gerência de workflows, permitindo a definição e execução dos mesmos. Todos eles possuem algum mecanismo para a definição abstrata de workflows e também, de alguma forma, executam esta definição. Nenhum deles, porém, fornece facilidades para a execução de apenas uma parte do experimento, aproveitando os dados de execuções passadas. Alguns desses sistemas também falham no registro das execuções de workflows e programas, ou ainda, por não utilizarem soluções abertas.

O projeto MyGrid tem basicamente as mesmas características dos projetos citados no parágrafo anterior, mas também possui um amplo suporte à descrição semântica de dados e programas através de ontologias. O projeto BioMoby, por outro lado, só fornece o suporte à descrição semântica, não tendo nenhuma pretensão quanto à gerência de workflows. Por último, a arquitetura SRMW, também combina a gerência de workflows a descrições semânticas de mais alto nível, através da utilização de metamodelos.

Outra característica interessante, é que todos os trabalhos, com exceção do LabBase/LabFlow, o mais antigo, estão adotando as tecnologias de serviços Web ou de Grids computacionais para a distribuição de seus projetos.

Tabela 1 - Tabela comparativa dos trabalhos relacionados

	LabBase/LabFlow	Chimera	MyGrid	Arsenal	gRNA	BioMoby	SRMW
(i) Definição abstrata do workflow	✓	✓	✓	✓	✓	×	✓
(ii) Definição do workflow em nível de programas executáveis	✓	✓	✓	✓	✓	×	×
(iii) Geração automática do workflow	✓	✓	×	×	✓	×	×
(iv) Execução do workflow	✓	✓	✓	✓	✓	×	×
(v) Re-execuções totais e parciais do workflow	×	×	✓	×	×	×	×
(vi) Tratamento de exceções na execução do workflow	×	×	✓	✓	×	×	×
(vii) Registro da execução de programas	✓	✓	✓	×	×	×	✓
(viii) Registro da execução de workflows	✓	×	✓	×	×	×	✓
(ix) Execução remota de programas utilizando padrões abertos	×	×	✓	✓	×	✓	✓
	LabBase	VDL	WS	WS	Jython	WS	WS
(x) Distribuição através da Internet	×	✓	✓	✓	✓	✓	✓
		Grid	Grid	WS	Grid	WS	WS

“Esfregou as mãos. Porque, veja bem, não se contentavam com incubar simplesmente os embriões: isso, qualquer vaca era capaz de fazer. Nós também predestinamos e condicionamos. Decantamos nossos bebês sob a forma de seres vivos socializados, sob a forma de Alfas ou de Ípsilons, de futuros carregadores ou de futuros Administradores Mundiais”.

Aldus Huxley. Admirável Mundo Novo

Uma vez apresentada a introdução inicial das tecnologias envolvidas nesta dissertação e analisadas as diversas iniciativas para o problema proposto, neste capítulo, será apresentada a arquitetura e a implementação do protótipo desenvolvido nesta dissertação, em conjunto com o exemplo de uso de uma aplicação real de bioinformática chamada de MHOLline. Na seção 4.1 será discutida a arquitetura do Ambiente 10+C e seus cinco principais componentes: as aplicações clientes, os serviços Web de aplicações científicas e de persistência de dados, e os módulos de definição e execução de workflows. Na seção 4.2 será visto os três tipos de usuários que interagem neste ambiente e, na seção 4.3, como ocorre esta interação durante o ciclo de vida de um experimento científico. Na seção 4.4, será apresentada uma aplicação real de bioinformática chamada MHOLline que foi utilizada como estudo de caso desta dissertação. Por fim, na seção 4.5, será visto como cada componente da arquitetura foi implementado utilizando uma aplicação Web escrita em Java, serviços Web e workflows científicos, com a utilização de exemplos do experimento MHOLline.

4.1 Arquitetura

A arquitetura do Ambiente 10+C de definição e execução de workflows científicos de serviços Web pode ser visualizado na Figura 17. Podemos dividir esta arquitetura em cinco componentes principais, a saber:

- (a) Aplicações Clientes
- (b) Serviços Web de Aplicações Científicas
- (c) Serviços Web de Persistência de Dados
- (d) Módulo de Definição de Workflows do Ambiente 10+C
- (e) Módulo de Execução de Workflows do Ambiente 10+C

O primeiro componente (Figura 17-a) contém as aplicações clientes através dos quais os usuários poderão utilizar os serviços disponíveis na arquitetura 10+C. Os clientes podem ser de duas maneiras distintas: navegadores Web que acessam os serviços via páginas HTML através do portal do ambiente 10+C ou aplicações que acessam diretamente via a interface de serviços Web, enviando e recebendo mensagens SOAP. Os serviços disponíveis para os usuários vão desde a definição e execução de um workflow, à execução de um programa isolado, até a navegação pelos resultados obtidos.

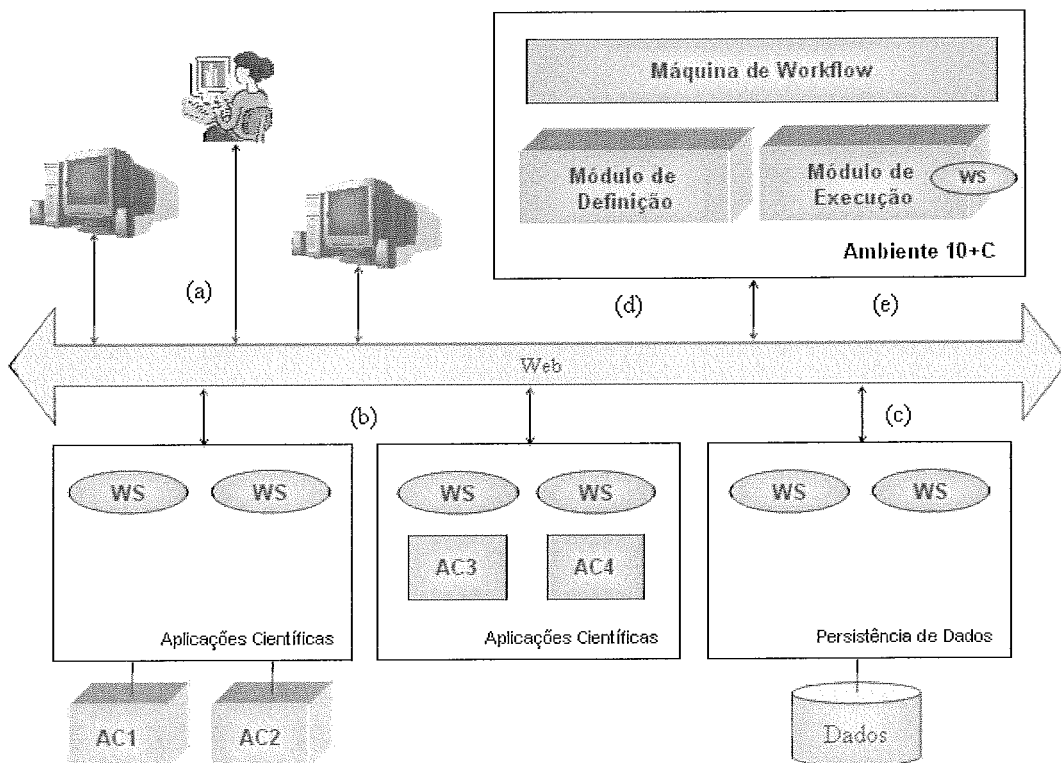


Figura 17 - Arquitetura do ambiente 10+C

O segundo componente da arquitetura (Figura 17-b) são representados pelas aplicações científicas (AC) que foram portadas para a tecnologia de serviços Web (WS). Essas aplicações podem ser programas legados escritos em linguagens de programação como Fortran ou C (AC1 e AC2) encapsuladas dentro de um serviço Web ou então aplicações escritas diretamente em linguagens que fornecem suporte nativo a serviços Web como Java ou Delphi (AC3 e AC4). De uma forma ou de outra, após a construção do serviço Web, o mesmo deverá ser instalado (*deploy*) em algum servidor de aplicação na Internet. Um servidor de aplicação tem basicamente o mesmo objetivo de um servidor de páginas Web, ou seja, prover conteúdo para os usuários. Porém o servidor de aplicação extrapola as funcionalidades oferecidas pelo anterior, pois permite o fornecimento de conteúdo dinâmico através do acesso a programas, componentes e banco de dados em oposição a páginas HTML estáticas. Ao instalar um serviço Web em um servidor de aplicação dizemos que o mesmo está publicado, já que agora ele poderá ser encontrado e utilizado por qualquer um.

Se forem utilizados programas legados, o serviço Web será apenas uma camada responsável em chamar o programa legado, passando os parâmetros que foram recebidos, resgatar os resultados e retorná-los aos clientes. Dessa forma, esses programas continuarão sendo executados da forma normal como foram projetados. Nas linguagens de programação mais modernas, o próprio programa já é uma aplicação Web que reside no servidor de aplicação, compartilha os seus recursos e é publicado automaticamente como um serviço Web.

Atualmente, no mundo dos negócios, cada vez mais, diversas aplicações estão sendo publicadas como serviços Web devido às inúmeras vantagens já apresentadas na Seção 3.1. Um exemplo disso é que diversos serviços de previsão de tempo [38][170] fornecem uma interface via serviços Web, onde basta informar o nome da cidade para conseguir um completo panorama do clima neste local.

Entretanto, na área de bioinformática, iniciativas de publicar programas científicos como serviços Web ainda são incipientes. A forma mais usual de distribuição desses aplicativos ainda é através de formulários HTML ou da disponibilização de executáveis locais. Por essa razão, fez-se necessário, como primeira implementação deste trabalho, publicar um conjunto de aplicações de bioinformática para serem utilizadas em workflows científicos, como será visto na Seção 4.5.

O terceiro componente da arquitetura (Figura 17-c) são representados pelos serviços Web que fornecem a persistência para os dados referentes às execuções de programas e workflows. Como exemplos de dados que podem persistir são os parâmetros utilizados, os dados de entrada, os resultados e os metadados referentes à execução. Essa funcionalidade é provida nesta arquitetura como um serviço independente, também implementado através de serviços Web. Dessa forma, na execução do workflow o usuário poderá informar se deseja fazer a persistência desses dados informando o endereço do serviço Web correspondente responsável. Neste protótipo, a persistência de dados é feita de forma simplificada armazenando os dados em arquivos texto em uma estrutura de diretório do próprio servidor de aplicação, como pode ser visto na Seção 4.5. Uma implementação mais sofisticada desta funcionalidade, que se preocupa em extrair os dados diretamente de mensagens SOAP e armazená-los de forma estruturada em um banco de dados, pode ser vista na dissertação de Mestrado de Teixeira [142] em andamento na COPPE/Sistemas.

O quarto componente (Figura 17-d) da arquitetura é o módulo de definição de workflows. Neste módulo, a partir dos serviços Web de aplicações científicas disponíveis, o usuário pode montar a seqüência de execução que deseja, concatenando as diferentes aplicações. Durante a definição do workflow, também será necessário efetuar o mapeamento de qual saída de dados de uma aplicação será utilizada como entrada em outra. Por fim, será gerado o arquivo de definição do workflow (um documento BPEL como será visto na Seção 4.5) que deverá posteriormente ser instalado (*deploy*) na máquina de workflow. Uma vez instalado, o workflow passa a estar publicado como um serviço Web e ficando disponível para ser encontrado e invocado por qualquer aplicação. Como cada workflow específico também é um serviço Web, pode-se inclusive reutilizá-lo na definição de outros workflows.

O último componente (Figura 17-e) representa o mecanismo de execução de workflows. Esse mecanismo é responsável em montar uma página HTML visual para a entrada de dados inicial do workflow e invocá-lo via sua interface de serviços Web. O workflow será executado na máquina de workflows do servidor de aplicação, invocando cada aplicação científica específica e passando os dados de entrada correspondentes. Durante a execução de cada aplicação, os dados referentes a essas execuções serão armazenados através dos serviços Web de persistência de dados para poderem ser utilizados em futuras re-execuções deste mesmo workflow.

4.2 Participantes da Arquitetura

O Ambiente 10+C descrito na seção anterior possui basicamente três tipos de usuários que irão interagir nesta arquitetura. O usuário publicador, o usuário que define o experimento e o usuário que irá executar o experimento. Esses usuários, na verdade, são papéis que alguma pessoa ou organização podem assumir, podendo existir a sobreposição de diversos papéis por uma mesma entidade.

O usuário publicador é o responsável por disponibilizar aplicações científicas e serviços de persistência para utilização da comunidade científica. Esses recursos são disponibilizados através de serviços Web. A idéia é que cada laboratório, quando desenvolver suas aplicações, também disponibilize uma interface via serviços Web, atuando como um provedor de serviços.

O definidor do experimento é o usuário que possui o conhecimento das aplicações, possui o conhecimento sobre o problema e deseja modelar um experimento para avaliar alguma hipótese de solução. Esse usuário é responsável por definir a sequência de execução do experimento a partir das aplicações disponíveis pelos provedores de serviço.

Finalmente o usuário final, é o usuário que vai executar um experimento definido anteriormente e avaliar os resultados obtidos, calibrando parâmetros e entradas até possuir alguma conclusão acerca do experimento. Este usuário também irá definir que nível de armazenamento de dados referente às execuções será necessário para cada experimento.

4.3 Ciclo de Vida

Uma vez descritos os componentes principais do Ambiente 10+C e os usuários que irão interagir com ele, nesta seção, será descrito o passo a passo de como um experimento científico é criado através de workflows de serviços Web nesta arquitetura.

O primeiro momento do ciclo de vida de um experimento científico acontece quando o experimento propriamente dito nem existe ainda. É a publicação dos programas na forma de serviços Web (Figura 18-a) pelo usuário publicador. Cada centro de pesquisa atua, desta forma, como um provedor de serviço oferecendo programas

como BLAST, FASTA e outros. Analogamente, provedores de serviço de armazenamento de dados, ou agentes de armazenamento, são responsáveis em publicar também via serviços Web (Figura 18-b), interfaces que provêm a persistência de determinados tipos de dados como, por exemplo, TEntradaBLAST, TSaidaBLAST, TEntradaFASTA e TSaidaFASTA, ou seja, os dados que serão usados/gerados pelos programas citados anteriormente. Note que um mesmo centro de pesquisa pode atuar como os dois tipos de prestadores de serviço. Na verdade, o ideal seria que uma vez que uma aplicação científica fosse disponibilizada, a sua interface que provê a persistência também seria. Para finalizar esta etapa, é necessário registrar o serviço Web no Ambiente 10+C para que o mesmo fique disponível para ser utilizado na definição de futuros workflows.

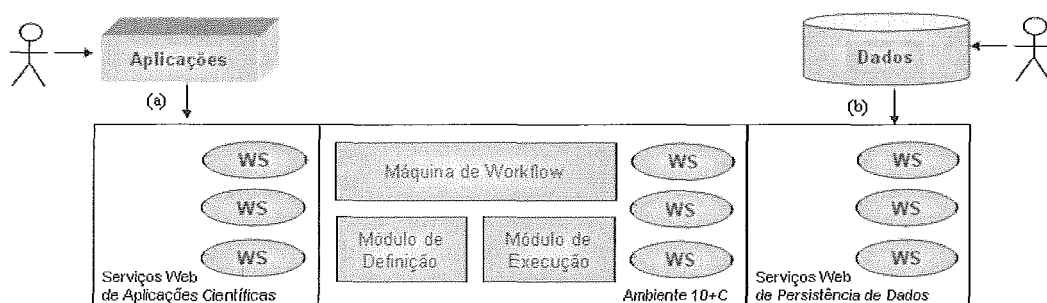


Figura 18 - Publicação de aplicações e interfaces de persistência de dados

Após a fase de publicação, a próxima etapa é a definição do workflow. Para isso, inicialmente é necessário especificar a definição da sequência de execução do mesmo. Essa especificação é feita pelo cientista que tem o conhecimento do problema em questão e que especificará o workflow correspondente a partir dos serviços Web disponíveis (Figura 19-c). O próprio portal do ambiente 10+C provê recursos para montar fluxos de execução sequencial dentre as aplicações científicas disponibilizadas no mesmo. Além disso, pode-se ainda, caso seja do interesse, utilizar alguma das diversas ferramentas de modelagem gráfica existentes atualmente no mercado, que fornecem recursos visuais sofisticados, ou até mesmo, um simples editor de texto como o bloco de notas, como será visto na seção 4.5.

Uma vez feita a especificação do workflow o próximo passo é a publicação do mesmo na máquina de execução. Caso a definição do workflow seja feita através do portal do Ambiente 10+C, os arquivos necessários para serem instalados na máquina de

workflow já são automaticamente gerados a partir da definição, cabendo ao usuário, apenas a tarefa de executar a publicação dos mesmos.

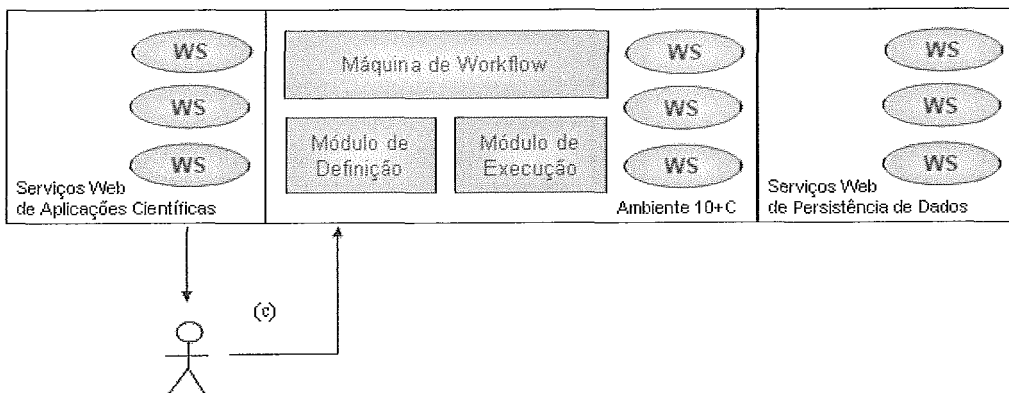


Figura 19 - Definição e publicação do workflow

A última fase deste ciclo de vida é a execução do experimento pelo usuário final, ou seja, um cientista ou biólogo que está estudando a aplicabilidade deste experimento. Esse workflow pode ser executado várias vezes pelo mesmo usuário, modificando parâmetros e dados de entrada, ou seja, criando-se diferentes instâncias do workflow. O usuário pode iniciar a execução do workflow de duas maneiras distintas: diretamente através do portal do Ambiente 10+C, onde ele escolherá qual workflow irá executar e o portal fica encarregado de montar uma página HTML para a entrada inicial de dados (Figura 20-d). Ou ainda, através do envio de mensagens SOAP diretamente para a máquina de execução, via a interface de serviços Web (Figura 20-e).

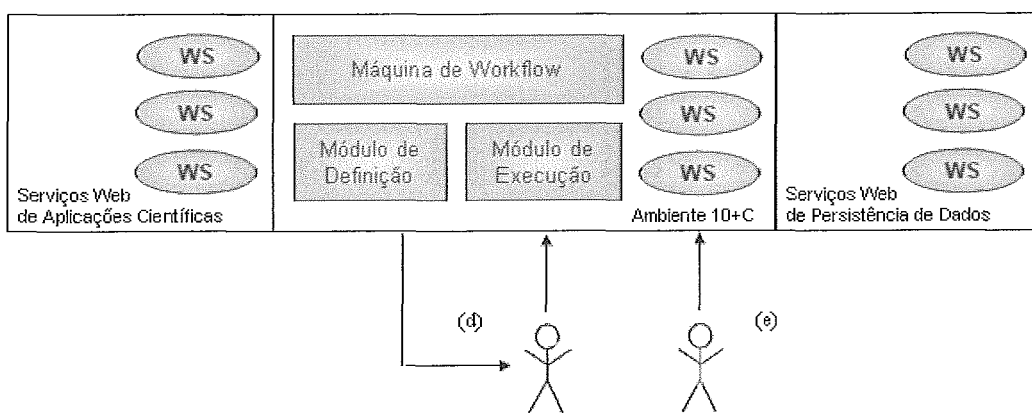


Figura 20 - Execução do Workflow

Uma vez iniciado o workflow, o mecanismo de execução é responsável por invocar cada serviço descrito na especificação (Figura 21-f), receber as respostas de cada um (Figura 21-g), passando os dados para o próximo seguindo a ordem correta.

Além disso, se foi especificado que os programas e o workflow teriam registro de execução, os dados intermediários seriam transferidos para o agente de persistência (Figura 21-h) para depois serem utilizados em possíveis execuções parciais de outras instâncias do mesmo workflow (Figura 21-i).

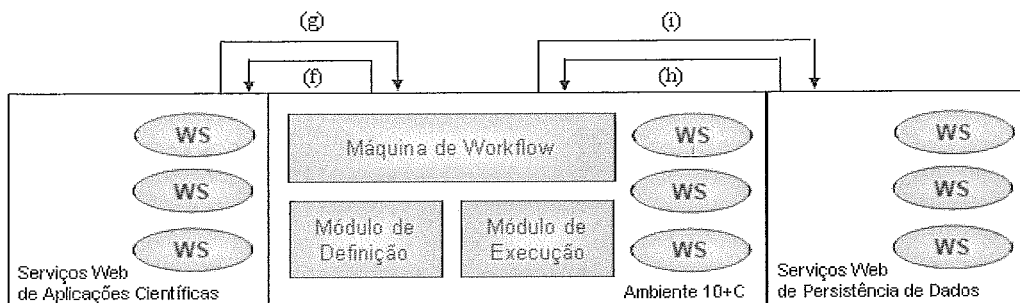


Figura 21 - Execução do Workflow

Todo este ciclo (Figura 22), principalmente a fase de execução do workflow, poderia ser repetido inúmeras vezes até um resultado plausível ser obtido pelo cientista. Na verdade, todo o ciclo é um processo interativo e incremental onde a cada execução, pode ser necessário fazer ajustes na própria definição do workflow, ou até mesmo, a criação de novas aplicações específicas para o problema pesquisado. De uma maneira ou de outra, o Ambiente 10+C é flexível suficiente para permitir as alterações necessárias às novas realidades das pesquisas.

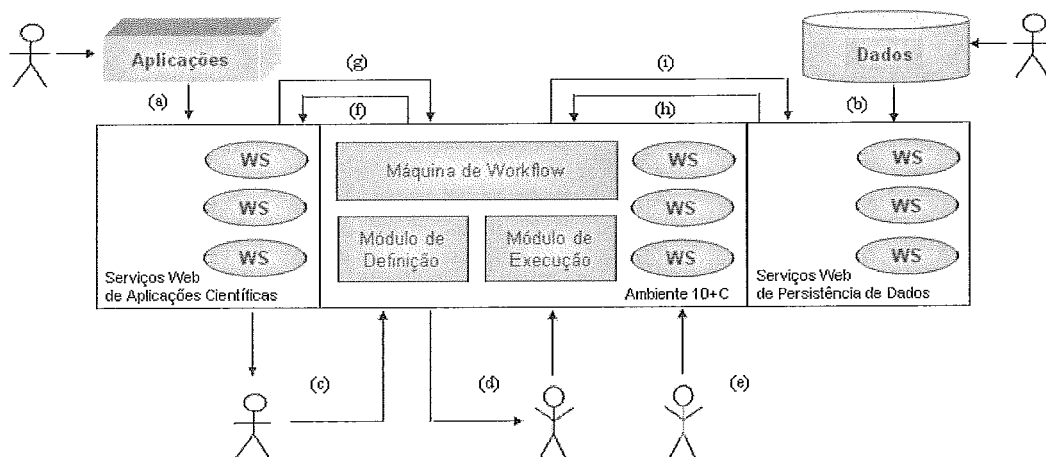


Figura 22 - Ciclo de vida completo de um experimento científico

O Ambiente 10+C e a SRMW

Como já foi comentado anteriormente em outras partes desta dissertação, a descrição em mais alto nível de programas, dados, serviços e workflows é vital para a composição de um ambiente robusto de experimentação científica. A gerência de

workflows no nível operacional (programa X da máquina Y) é essencial para a produção de experimentos científicos, mas a categorização e classificação dos metadados referentes a estes recursos podem fornecer uma ampla gama de possibilidades adicionais aos biólogos e cientistas. Perguntas como: 1) A qual categoria a aplicação faz parte? 2) Qual classificação? 3) Quais categorias de dados são utilizados? 4) Quais são os programas semelhantes? Entre outras, podem ser mais facilmente respondidas com a utilização de um ambiente de gerência de metadados.

Neste contexto, caso seja necessário o suporte de todo esse ferramental semântico na experimentação científica, o Ambiente 10+C poderia ser utilizado em conjunto com a arquitetura SRMW para prover o casamento do nível meta com o nível operacional, habilitando todo o poder de um laboratório virtual de experimentação científica. Uma prova de conceito dessa integração, utilizando um protótipo simplificado do Ambiente 10+C, foi realizado através do experimento científico MHOLline, conforme relatado por Cavalcanti, Targino *et al.* [44].

A SRMW pode ser utilizada durante todo o ciclo de vida de um experimento no Ambiente 10+C, desde a publicação dos programas científicos, até a definição e execução dos workflows. Tarefas como a publicação de uma aplicação científica ou a definição de um novo workflow no Ambiente 10+C seriam complementadas com a respectiva publicação de uma descrição mais precisa na arquitetura SRMW. A definição do workflow pelo cientista ou biólogo no Ambiente 10+C, também seria facilitada pela utilização das informações adicionais sobre as aplicações científicas e sobre os dados, existentes na arquitetura SRMW.

Na execução do experimento, o cientista ou biólogo pode primeiramente navegar sobre os metadados da arquitetura SRMW a fim de encontrar qual experimento já publicado melhor se adequa ao objetivo pretendido. Uma vez escolhido o experimento, basta ao usuário efetuar a execução do mesmo, através do Ambiente 10+C.

4.4 O Workflow MHOLline

O MHOLline [131] é um exemplo de um experimento *in silico* que utiliza diversas aplicações para atingir seus objetivos. Projetado por Rössle [130] e desenvolvido no instituto de Biofísica da UFRJ (IBCCF/UFRJ), tem como objetivo a

predição e construção de modelos tridimensionais de proteínas a partir de uma seqüência de aminoácidos. Este tipo de experimento tem grande importância na descoberta das funções dos genes dos organismos.

O MHOLline é composto por um conjunto de programas de bioinformática de código aberto disponibilizados na Internet e mais uma aplicação desenvolvida por Rössle [131] (Figura 23). Esta aplicação, BATS (*Blast Automatic Targeting for Structures*), é um filtro que escolhe as seqüências que serão utilizadas como base para a descoberta da estrutura da proteína pesquisada e é fundamental na automatização do workflow. A operacionalização do workflow é feita através de um *script* escrito na linguagem Perl, responsável pela invocação de cada um dos programas participantes do experimento, seguindo uma ordem de execução pré-estabelecida e passando os dados entre eles.

A entrada inicial do experimento é um arquivo de seqüências de aminoácidos no formato FASTA que após a execução do MHOLline gera um arquivo contendo as coordenadas x, y e z de cada molécula da proteína.

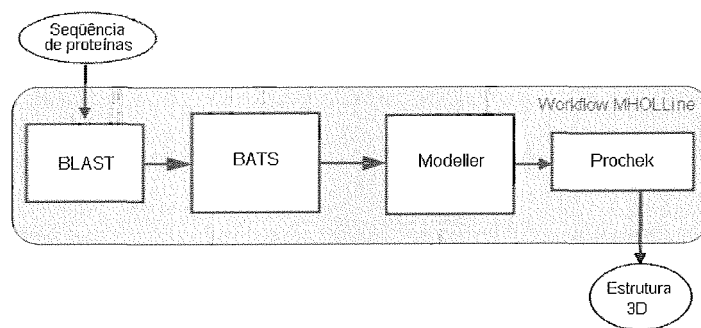


Figura 23 - Workflow MHOLline

O primeiro programa a ser executado neste experimento é o BLAST, em sua versão blastp do NCBI, executado localmente. A partir do arquivo FASTA de entrada, é obtido através de comparações no banco de dados PDB, um conjunto de seqüências de proteínas que possuem similaridade local seqüencial à seqüência submetida. As seqüências que possuem mais similaridade, chamadas de referência, serão utilizadas como molde para a construção do modelo tridimensional.

O resultado do BLAST pode apresentar diversas seqüências de referências, mas apenas aquelas que satisfazem um mínimo de pré-requisitos poderão ser utilizadas como molde. Para isso, foi desenvolvido o programa BATS que é responsável em calcular

parâmetros de qualidade para as referências e filtrar os resultados a partir de um limite pré-estabelecido. Esse programa também foi escrito na linguagem Perl e, na verdade, está embutido no mesmo *script* responsável pela execução do workflow.

O BATS também é responsável em montar os arquivos de entrada que serão utilizados no próximo programa do experimento: o Modeller. O Modeller é o programa que irá criar a estrutura tridimensional a partir da referência encontrada nas etapas anteriores efetuando uma série de cálculos complexos. Este é o programa que mais leva tempo e consome recursos computacionais deste workflow. Por fim, o programa PROCHEK é executado para validar se o resultado obtido possui uma estrutura fisicamente estável, ou seja, é uma estrutura que pode ocorrer na natureza.

O conjunto desses programas, denominado MHOLline, na verdade, é apenas uma das diversas seqüências possíveis de execução do MHOLline. Após a execução do BATS, outros dois caminhos poderiam ser percorridos conforme o resultado deste programa. Para esta dissertação, simplificamos o workflow do MHOLline fixando o seu caminho de execução principal.

A grande contribuição do MHOLline foi combinar diversas aplicações científicas disponíveis na comunidade em uma seqüência de execução executável para fornecer um resultado confiável, que no caso deste experimento, são as estruturas tridimensionais. Grande parte do sucesso desta combinação se deve ao próprio programa escrito no IBCCF/UFRJ, o BATS, que é o responsável por analisar os resultados iniciais do BLAST e direcionar o MHOLline para a seqüência de execução mais apropriada. Através da utilização do experimento MHOLline, cientistas menos experientes podem efetivamente participar da pesquisa de descoberta de estruturas de proteínas, já que o MHOLline já possui o ferramental necessário. Além disso, o programa BLAST já executa uma análise preliminar dos resultados, direcionando a pesquisa para o caminho mais adequado. Porém, como veremos na seção 5.2, a utilização de linguagens de *scripts* para este tipo de experimento, limita enormemente o potencial de pesquisa do biólogo ou centro de pesquisa.

4.5 Implementação do MHOLline no Ambiente 10+C

Em toda a implementação dos componentes da arquitetura do Ambiente 10+C foram utilizadas linguagens e ferramentas de código aberto disponíveis na Internet. Essa

foi uma das maiores preocupações no desenvolvimento deste ambiente, pois era de grande importância que a solução não ficasse amarrada em nenhum tipo de tecnologia proprietária. A utilização de ferramentas e padrões abertos é fundamental para a disseminação e utilização de qualquer solução em ambientes heterogêneos nos dias de hoje.

Todos os programas foram escritos na linguagem Java [91] que atualmente é uma das linguagens mais utilizadas e tem se tornado um padrão em projetos de código aberto. A variedade de ferramentas disponíveis e a existência de uma comunidade ativa de desenvolvedores certificam-na para utilização em qualquer tipo de projeto, seja de pesquisa ou comercial.

O servidor de aplicação Apache Tomcat 4.1.18 [12] foi utilizado para disponibilizar os serviços Web. Apesar de não ser próprio para um ambiente de produção, este servidor atendeu aos propósitos deste trabalho e no futuro, se fosse necessário, poderia ser integrado ao servidor Apache [11] para prover um ambiente mais robusto.

A este servidor, foi necessário adicionar o pacote AXIS [15] que é responsável em disponibilizar classes Java como serviços Web e em atender as requisições de mensagens SOAP. Para a execução dos workflows, foi escolhida a linguagem de definição BPEL para especificar a seqüência de execução. Essa linguagem possui um mecanismo de execução, chamado de BPWS4J 1.0.1, feito pela IBM [32] que também é livremente distribuído e foi utilizado neste trabalho.

Também foi necessário utilizar um sistema gerenciador de banco de dados para armazenar as informações referentes à especificação dos workflows e às informações de cada serviço Web disponível no Ambiente 10+C. O servidor MySQL 4.0.20 [111] foi escolhido para efetuar a persistência desses dados.

Nesta seção, iremos descrever a implementação de cada um dos cinco componentes definidos na seção 4.1 utilizando exemplos do experimento MHOLline.

Aplicações Cliente

Para a criação e o acompanhamento de experimentos científicos de bioinformática foi criado um portal na Web que fornece o suporte as atividades de definição, execução, utilização e visualização de workflows e serviços Web para os três tipos de usuários da arquitetura (Publicador, Definidor de Experimento e Usuário Final).

O portal do Ambiente 10+C consiste de um conjunto de páginas HTML e classes *servlets* Java que fornecem uma interface visual (Figura 24) para atividades como: registrar aplicações científicas, criar e gerar novos workflows, executar serviços Web e workflows, além de visualizar aplicações científicas e workflows disponíveis no ambiente.

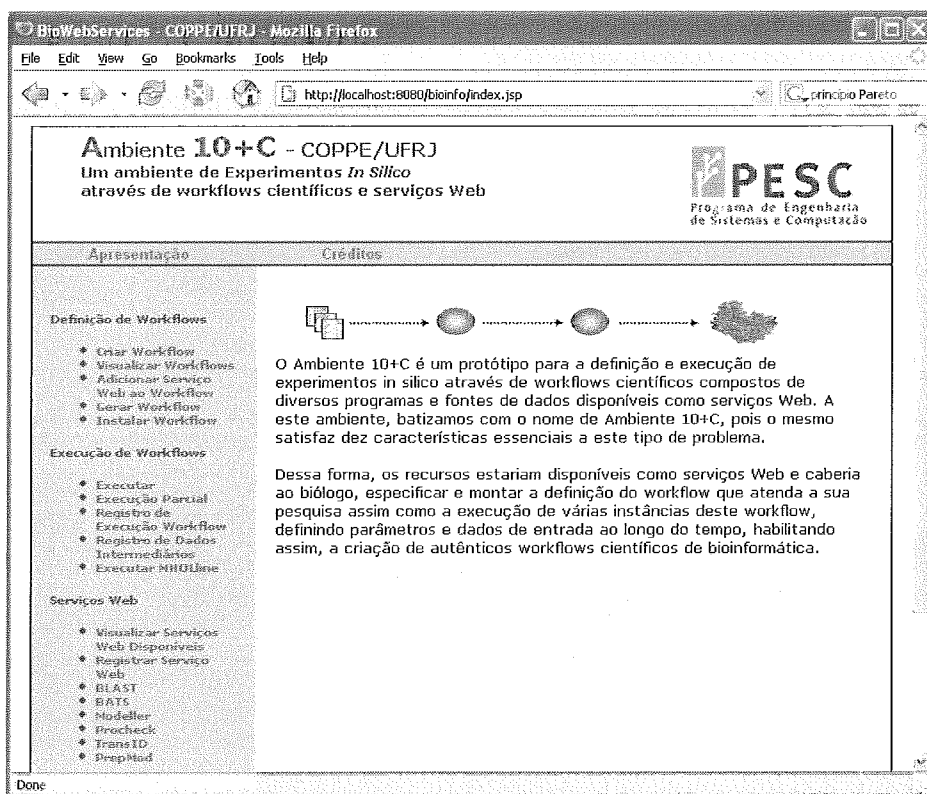


Figura 24 - Portal do Ambiente 10+C

Outra forma de execução dos workflows científicos pode ser feita através do envio direto de mensagens SOAP para o servidor de aplicação, já que cada workflow também é publicado como um serviço Web. Basta dessa forma, enviar uma mensagem SOAP diretamente ao serviço através de uma URL (*Unified Resource Location*) como <http://192.168.0.15/bioinfo/mhonline.jsv?wsdl> empacotando a operação que será executada com todos os parâmetros de entrada. Essa mensagem é processada no

servidor de aplicação pelo módulo AXIS, o serviço Web correspondente é executado e o resultado é retornado através de outra mensagem SOAP.

Serviços Web de Aplicações Científicas

Aplicações científicas já disponibilizadas como serviços Web precisam apenas ser registradas no Ambiente 10+C, para que as mesmas fiquem disponíveis para serem utilizadas na definição de qualquer workflow (Figura 25). O registro de serviços Web no Ambiente 10+C é necessário para identificar o endereço e todas as entradas e saídas de cada aplicação. Se a definição de workflows for feita externamente ao ambiente, este procedimento não será necessário.

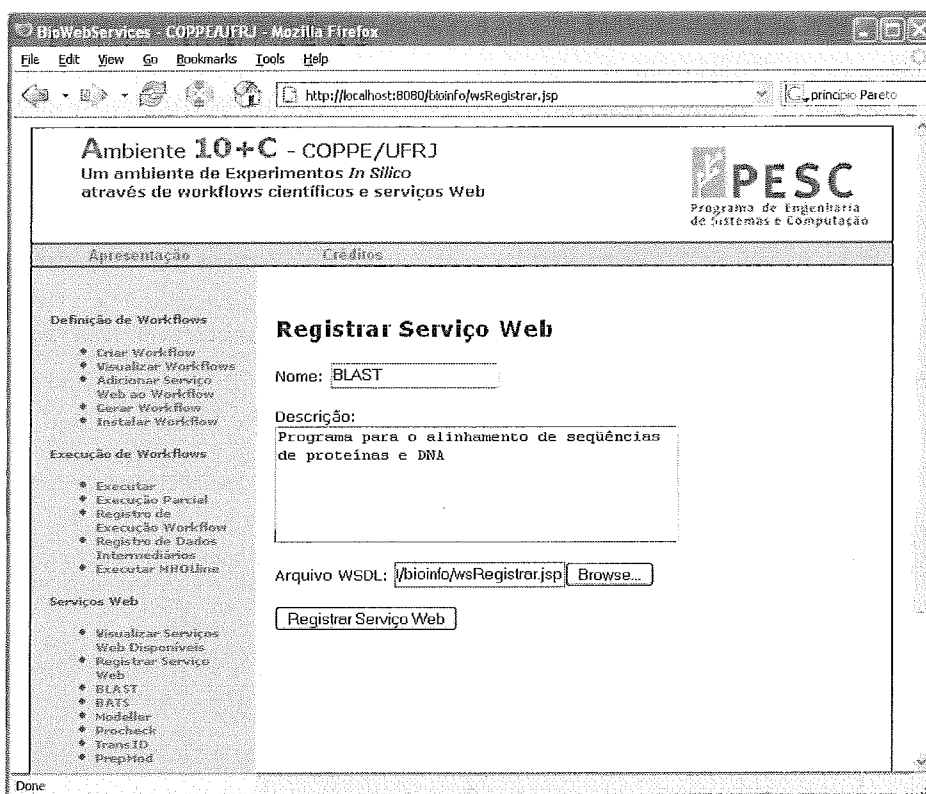


Figura 25 -- Registro de serviços Web no Ambiente 10+C

No caso do workflow MHOLline, inicialmente, nenhum de seus programas (BLAST, BATS, Modeller e Procheck) estavam disponíveis como serviços Web. O desafio, portanto, foi transformar os programas usados pelo MHOLline, que são proprietários e não possuem nem código fonte disponível, nem uma API de acesso definida, em serviços Web acessíveis na Internet e com uma interface estruturada e bem conhecida.

Para isso, foram desenvolvidos programas tradutores (*wrappers*) que atuavam como uma camada intermediária responsável em exportar a interface do serviço para o mundo externo e acessar as aplicações legadas. Esses tradutores foram escritos utilizando a linguagem Java e tinham como função básica converter a entrada de dados para um formato correspondente ao programa utilizado, executá-lo e depois ler o resultado e retornar a saída (Figura 26).

```

public class BlastRunner {
    public String[] runBlast(String[] seqFasta, String BlastType, String TargetDB, String
OutputType)
    ...
    params = "-p " + BlastType + " -d " + TargetDB
        + " -i " + fileID + "IN.txt" + " -o "
        + fileID + "OUT.txt" + " -m " + OutputType;
    processName = "C:\\blast\\blastall" + params;
    runtime = Runtime.getRuntime();
    myProcess = runtime.exec(processName);
    ...
}

```

Figura 26 – Tradutor para o programa Blast

Uma vez criado os programas tradutores, o próximo passo foi instalá-los no servidor de aplicação, utilizando o pacote AXIS para transformar as classes Java em serviços Web instalados e prontos para invocação por outras aplicações, como o próprio workflow do MHOLline (Figura 27).

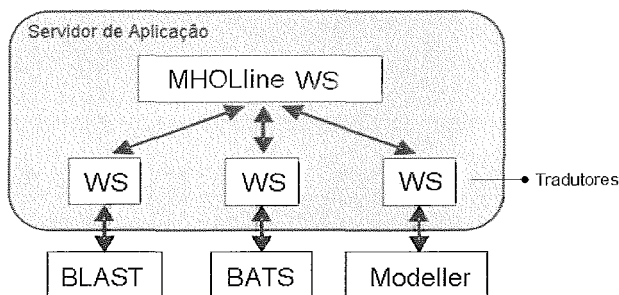


Figura 27 - Programas científicos disponíveis como serviços Web

A Figura 28 mostra o WSDL do serviço Web correspondente ao programa BLAST. Note que o código possui uma porta de entrada BlastRunner que possui uma operação RunBlast a partir da qual é possível o envio de mensagens SOAP de entrada runBlastRequest e recebimento das mensagens SOAP de saída runBlastResponse.

```

<wsdl:message name="runBlastRequest">
    <wsdl:part name="SeqFasta" type="xsd:string"/>
    <wsdl:partname="BlastType" type="xsd:string"/>
    <wsdl:part name="TargetDB" type="xsd:string"/>
    <wsdl:part name="OutputType" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="runBlastResponse">
    <wsdl:part name="runBlastReturn" type="xsd:string"/>

```

```

</wsdl:message>
<wsdl:portType name="BlastRunner">
  <wsdl:operation name="runBlast">
    <wsdl:input message="intf:runBlastRequest"/>
    <wsdl:output message="intf:runBlastResponse"/>
  </wsdl:operation>
</wsdl:portType>

```

Figura 28 - Arquivo WSDL do BLAST

Os programas BATS, Modeller e Procheck tiveram a mesma estrutura de implementação do programa anterior. A diferença básica é que o número e tipo dos parâmetros de entrada e saída são diferentes precisando da construção de um tradutor específico para cada um desses programas. Uma vez disponibilizados como serviços Web, esses programas foram registrados no Ambiente 10+C como visto na Figura 25.

Serviços Web de Persistência de Dados

Após a execução de cada um dos serviços Web, os dados resultantes podem ser armazenados para futuras reutilizações e análises. Se esta opção for escolhida na execução do workflow, o agente de persistência de dados será acionado para efetuar a persistência dos mesmos.

Esse agente de armazenamento de dados, na verdade é um outro serviço Web que recebe os dados como parâmetros e os armazena em um esquema estruturado e genérico no banco de dados conforme os tipos definidos na mensagem SOAP. Esta funcionalidade está em andamento na COPPE/Sistemas no trabalho de dissertação de mestrado de Teixeira [142].

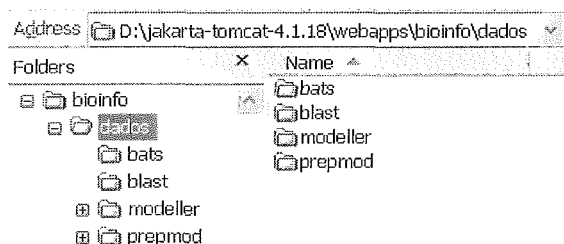


Figura 29 - Armazenamento de dados intermediários no servidor de aplicação

Neste trabalho, foi adotado um mecanismo de persistência simplificado encarregado de armazenar os dados intermediários como arquivos textos na própria estrutura de diretórios do servidor de aplicações (Figura 29). Esse mecanismo foi construído através de serviços Web e era invocado pelo workflow para armazenar o resultado de cada um dos programas executados. Esse serviço recebia como entrada de dados, um número identificador de execução do workflow, chamado de *TransID*, além

do próprio dado a ser armazenado. O número identificador TransID será discutido mais a frente nesta seção.

Na definição do workflow é necessário especificar os serviços Web de persistência de dados, da mesma maneira como fazemos para qualquer outro serviço. Logo após a execução do serviço Web do BLAST, por exemplo, será necessário invocar o serviço Web de persistência de dados do BLAST, para armazenar os resultados obtidos.

Da mesma maneira, foram criados serviços Web para efetuar a recuperação destes dados. Essa recuperação é feita utilizando o número TransID, que é o identificador do dado na estrutura de diretório do servidor de aplicação. Os serviços Web de recuperação de dados são utilizados nas re-execuções parciais de workflows, quando parte da seqüência de execução não precisa ser novamente executada. É possível ainda, visualizar os dados armazenados, através do portal do Ambiente 10+C, navegando pela estrutura de diretórios do servidor (Figura 30).

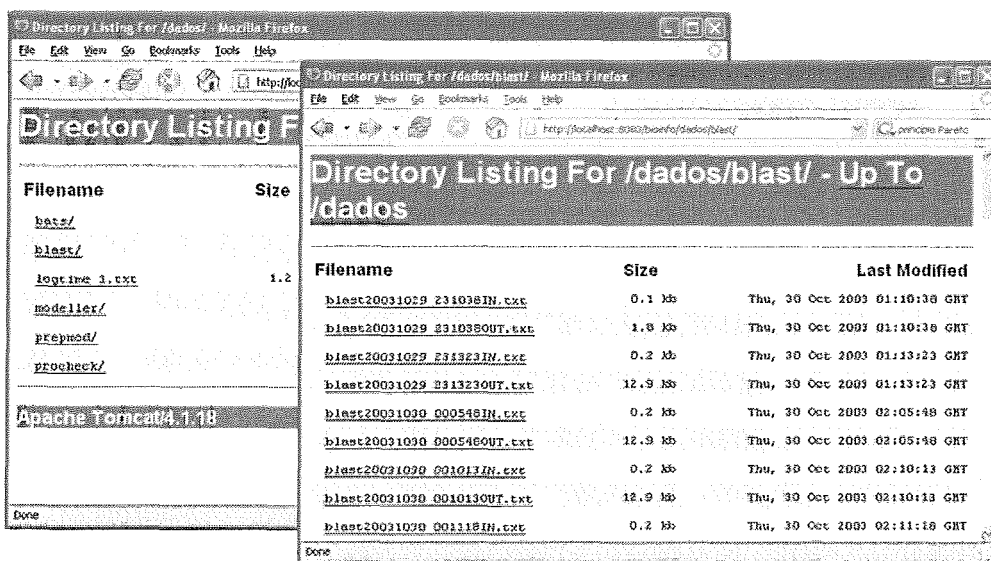


Figura 30 - Dados intermediários de execuções dos workflows

Módulo de Definição de Workflows

Uma vez disponibilizados os serviços Web do MHOLine, a definição do workflow pelo cientista pode ser executada através do portal do Ambiente 10+C. Pelo portal, conforme pode ser visto na Figura 31, existem os recursos necessários para permitir a criação, escolha dos passos e geração do workflow desejado.

Para criar um novo workflow, o usuário, inicialmente, informará o seu nome e a sua descrição (Figura 31-1). Depois, deverá escolher, entre os serviços Web previamente registrados, cada uma das aplicações científicas do seu experimento na ordem de execução desejada (Figura 31-2). Entre uma aplicação científica e outra, ele deverá especificar a entrada de cada uma. Para isso, ele pode escolher entre as saídas de aplicações anteriores da seqüência de execução, ou então, indicar que este parâmetro vem diretamente como entrada de dados do próprio workflow (Figura 31-3). Apenas a criação de workflows seqüenciais são atualmente suportados pelo Ambiente 10+C.

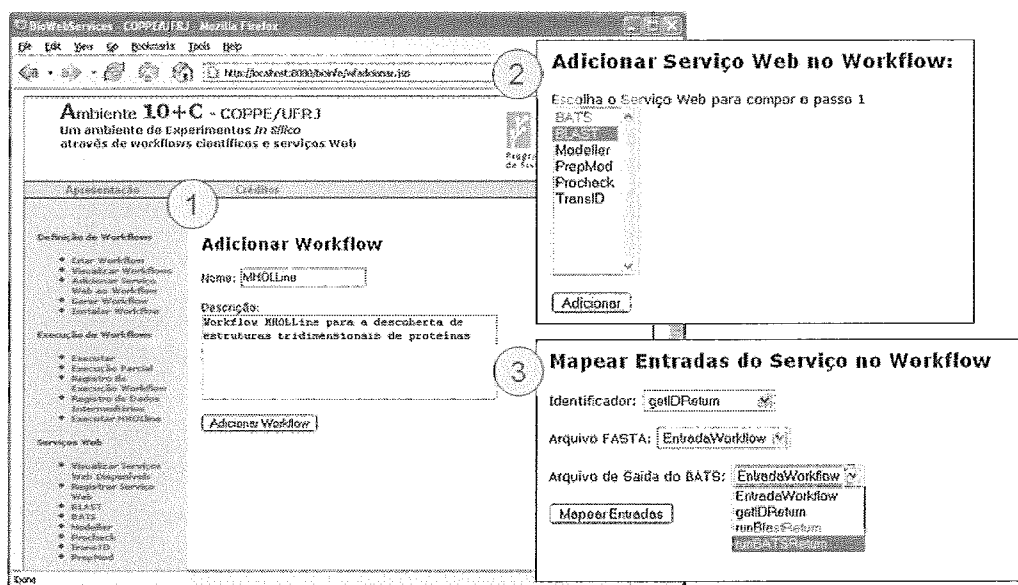


Figura 31 - Criação de workflows pelo portal do Ambiente 10+C

Toda definição do novo workflow e os mapeamentos dos parâmetros de entradas de cada serviço são armazenados no banco de dados MySQL em um esquema com apenas cinco tabelas: Workflow, Serviços, Variável, Passo_Workflow e Entrada (Figura 32). Finalizada esta etapa (Figura 33), o usuário estará apto a gerar os arquivos de especificação do workflow propriamente dito. Esses arquivos foram gerados a partir dessas tabelas por um *servlet* Java hospedado no servidor de aplicação.

Os arquivos de especificação dos workflows foram efetuados utilizando uma linguagem de definição de processos de workflows. Das diversas propostas existentes atualmente, foi escolhida a linguagem de definição BPEL4WS – *Business Process Execution Language for Web Services* [52], ou abreviadamente BPEL, patrocinada por grandes fornecedores como IBM, Microsoft e BEA. O BPEL é uma linguagem baseada em XML para a coordenação de processos sobre a Internet, e constitui-se atualmente,

em um forte candidato a ser transformado em um padrão de fato para composição de serviços Web.

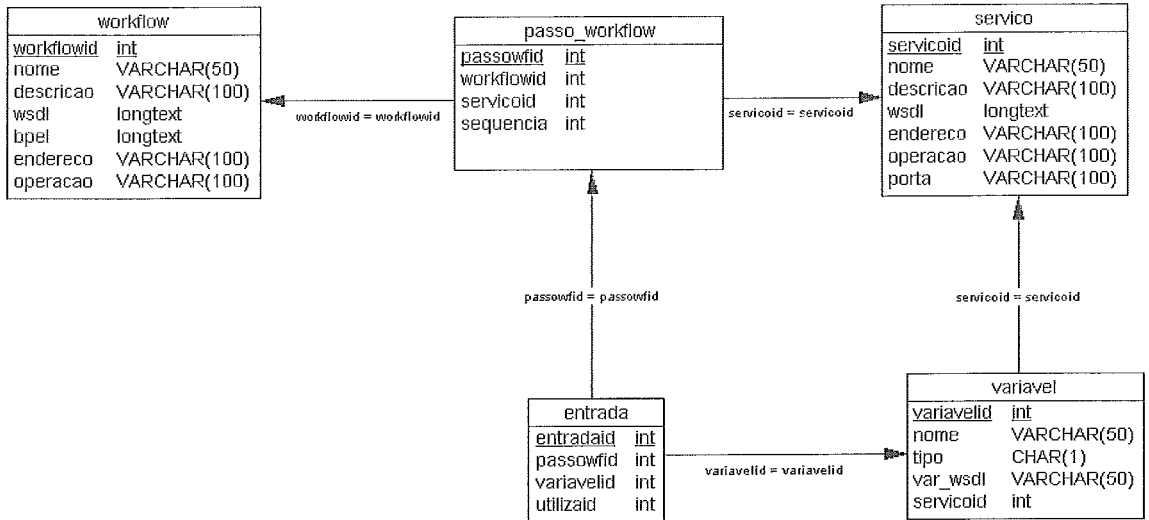


Figura 32 - Modelo de Dados do Ambiente 10+C

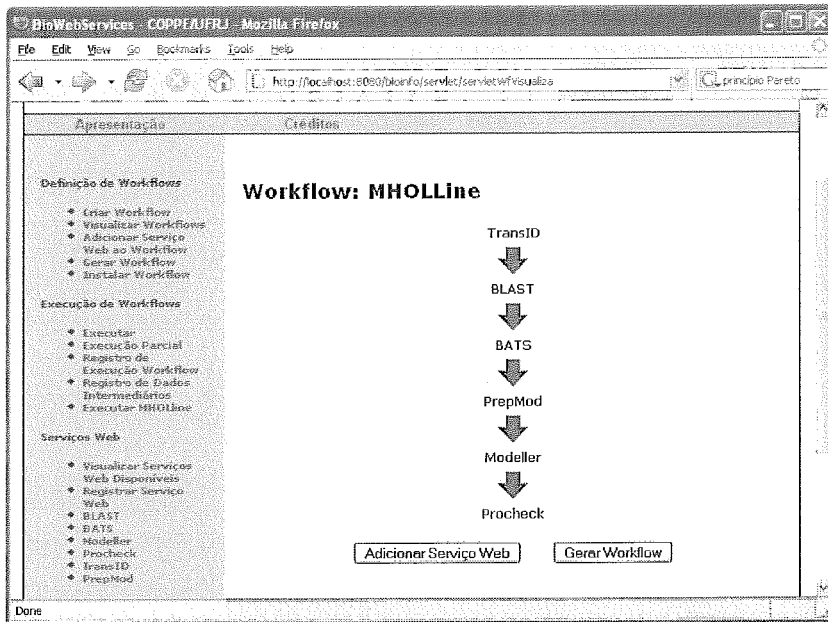


Figura 33 - Visualização do Workflow MHOLline no Ambiente 10+C

O BPEL possui diversos construtores que são utilizados para manipular as tarefas, os dados, a ordem de execução e o mapeamento de dados entre os processos durante cada etapa do workflow. Esses construtores representam desde o recebimento de dados pelo workflow até a invocação de um serviço Web do mesmo. Podemos dividir os construtores do BPEL em dois grupos. Os construtores específicos para definição dos componentes participantes dos workflows como <partner> e <container> e os construtores relacionados ao fluxo de execução do workflow, como <sequence>,

<receive>, <invoke>, <wait> e <reply>. Cada etapa do workflow é chamada em BPEL de uma atividade.

A composição dos programas é definida em nível mais alto como um processo <process>. Cada programa do workflow é definido como um parceiro <partner>. No workflow MHOLline, os programas BLAST, BATS, Modeller e Procheck, além do próprio usuário do workflow (o *caller*), são definidos como parceiros, como mostra a Figura 34. Existe também um construtor específico para armazenar os dados enviados e recebidos por cada atividade, chamado de <container>. Esse construtor atua da mesma forma que uma variável nas linguagens de programação, permitindo que os dados sejam passados entre uma atividade e outra ou então analisados para um possível desvio na ordem de execução. Como a entrada de um programa é normalmente diferente da sua saída, são utilizados dois <container> para cada serviço Web a ser invocado, como os <container name="blastrequest"> e <container name="blastresponse"> na Figura 34.

```
<partners>
  <partner name="caller"
    serviceLinkType="tns:runnerSLT"/>
  <partner name="providerblast"
    serviceLinkType="blastns:blastSLT"/>
  <partner name="providerbats"
    serviceLinkType="batsns:batsSLT"/>
  <partner name="providermodeller"
    serviceLinkType="modns:modellerSLT"/>
  <partner name="providerprocheck"
    serviceLinkType="modns:procheckSLT"/>
</partners>

<containers>
  <container name="request"
    messageType="tns:request"/>
  <container name="response"
    messageType="tns:response"/>
  <container name="blastrequest"
    messageType="blastns:runBlastRequest"/>
  <container name="blastresponse"
    messageType="blastns:runBlastResponse"/>
  <container name="batsrequest"
    messageType="batsns:runBATSRequest"/>
  ...
</containers>
```

Figura 34 - Participantes do workflow MHOLline escrito em BPEL

A composição das atividades do workflow propriamente dito, é implementada através do comando <sequence> no qual se permite definir uma ordem de execução dos serviços oferecidos pelos parceiros do processo. Entre os comandos que podem aparecer dentro de <sequence>, destacam-se o <receive> para a entrada de dados quando o processo é invocado, o <invoke> que permite a execução de serviços Web

externos (dos parceiros) e `<reply>` que retorna ao invocador a mensagem de saída. Para fluxos de execução não sequencial pode-se utilizar o comando `<flow>` ao invés de `<sequence>`.

O primeiro passo do processo é receber os dados de entrada do usuário que está iniciando o workflow através do comando `<receive>` (Figura 35-A). Esse construtor é específico para representar a chegada dos dados de entrada do workflow. Parte desses dados são parâmetros de entrada do serviço BLAST e serão repassados para ele, posteriormente, através do construtor `<assign>` (Figura 35-B). O construtor `<assign>` é utilizado para copiar os dados de um `<container>` para outro. A invocação de serviços como o BLAST, é feita através do comando `<invoke>` e passando o `<container name="blastrequest">` previamente definido como parâmetro, como mostra a Figura 35-C. Por fim, o resultado final é retornado para o usuário através do construtor `<reply>` (Figura 35-D).

```

<sequence>
  <receive name="receive" partner="caller"
    portType="tns:runnerPT" operation="runnerOP"
    container="request" createInstance="yes"/> (A)

  <assign name="requestToServiceblast">
    <copy><from container="request" part="in1"/>
      <to container="blastrequest" part="SeqFasta"/></copy>
    <copy><from container="request" part="in2"/>
      <to container="blastrequest" part="BlastType"/></copy> (B)
    <copy><from container="request" part="in3"/>
      <to container="blastrequest" part="TargetDB"/></copy>
    <copy><from container="request" part="in4"/>
      <to container="blastrequest" part="OutputType"/></copy>
  </assign>

  <invoke name="invoke" partner="providerblast"
    portType="blastns:BlastRunner" operation="runBlast"
    inputContainer="blastrequest" outputContainer="blastresponse"/> (C)

  (...)

  <reply name="reply" partner="caller"
    portType="tns:runnerPT"
    operation="runnerOP"
    container="response"/> (D)

</sequence>

```

Figura 35 – Construtor `<receive>`

Após a geração do workflow (Figura 36-1), os arquivos BPEL e WSDL deste experimento serão gerados e deverão ser instalados na máquina de execução de workflow (Figura 36-2). Neste processo, os WSDL de cada serviço que o workflow invoca também deverão ser informados. Após esta instalação, o workflow estará publicado como um serviço Web e prontamente disponível para uso.

Para a criação de workflows mais complexos, com testes e desvios no fluxo de execução, pode-se criar o arquivo BPEL manualmente, através de editores de texto de XML ou ferramentas gráficas de construção de workflows. A utilização de editores de texto de XML como o XML Spy [7] ou até mesmo o *notepad* não traz nenhuma facilidade na criação do arquivo BPEL, sendo necessário o conhecimento dos construtores da linguagem BPEL.

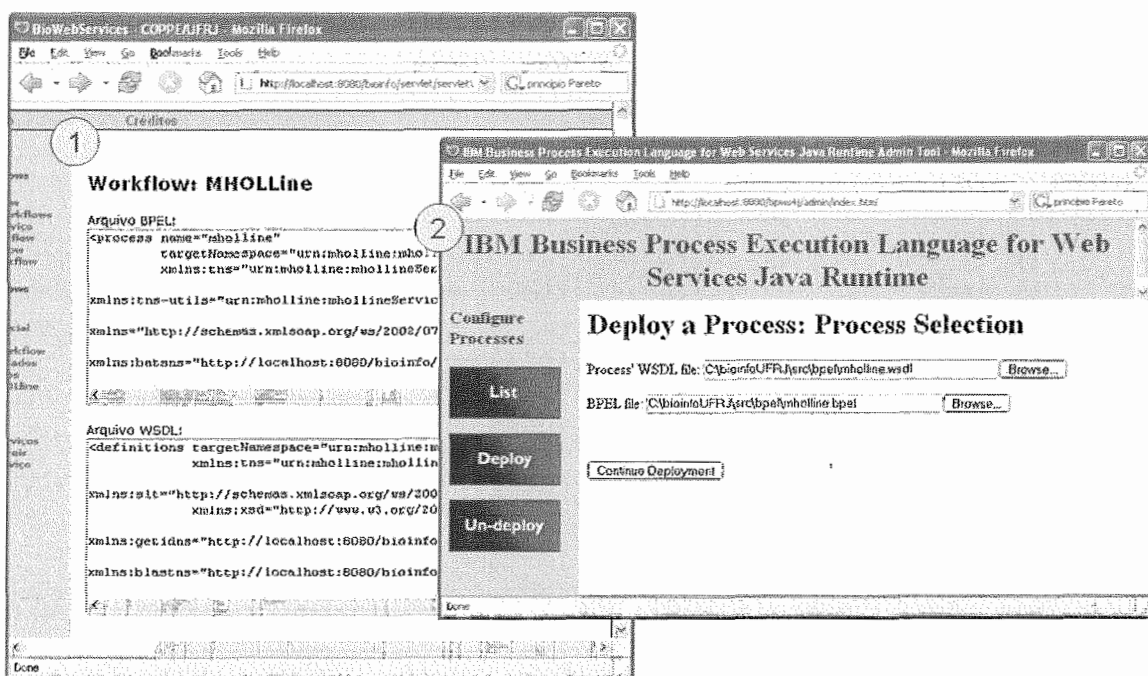


Figura 36 - Geração e instalação da especificação do workflow

A utilização de ferramentas gráficas para a definição dos arquivos BPEL pode simplificar o processo de especificação do fluxo de execução e permitir que qualquer tipo de usuário, sem maiores esforços, execute este tipo de tarefa. Ferramentas comerciais como o Visual Script [159] (Figura 37-1) ou ferramentas livres como *plugin* para o ambiente de Eclipse [59] (Figura 37-2), fornecem, através de diagramas e notações gráficas, os elementos necessários para a criação de qualquer tipo de experimento.

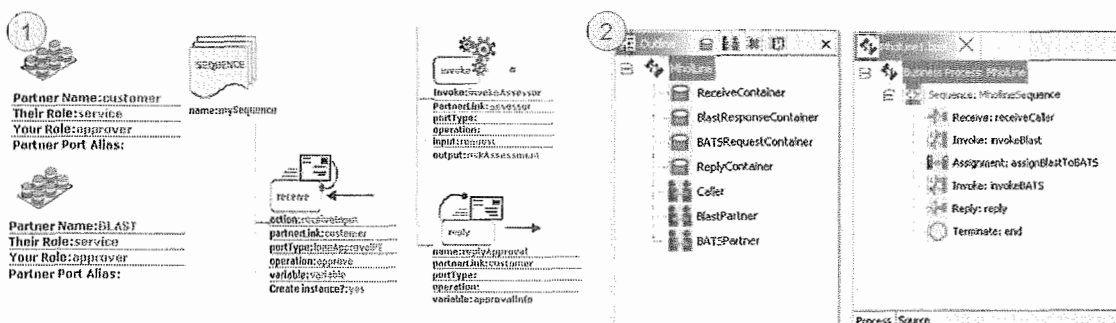


Figura 37 – Exemplo hipotético de definição de workflow com o Visual Script [159] e Eclipse [59]

Módulo de Execução de Workflows

Para a execução dos workflows, foi desenvolvido no Ambiente 10+C um *servlet* responsável em montar uma página HTML para a entrada de dados do workflow (Figura 38) a partir da definição armazenada no banco de dados MySQL. Após o preenchimento das entradas por parte do usuário, a página invoca um *proxy* genérico que é uma classe especializada em montar mensagens SOAP para serem enviadas e recebê-las de volta obtendo os dados resultantes.

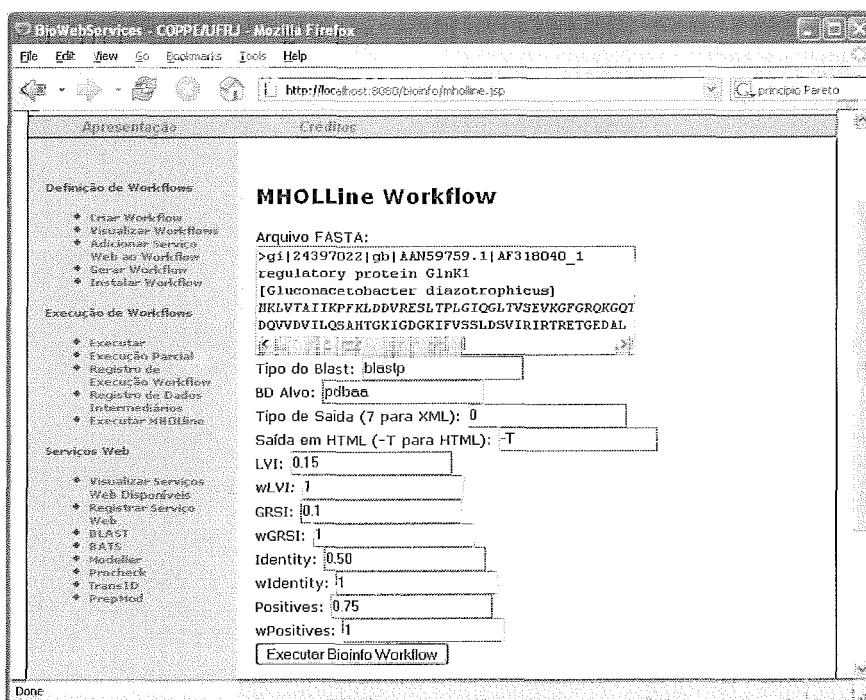


Figura 38 - Entrada de dados do workflow MHOline

Para o workflow MHOline, a entrada principal é uma seqüência de proteínas no formato FASTA como pode ser visto na Figura 38. As entradas iniciais dos programas BLAST e BATS, que não são obtidas a partir de nenhum outro programa, também deverão constar como entrada do próprio workflow MHOline. A seqüência de proteínas de entrada é primeiramente repassada ao BLAST através do construtor <assign>. O resultado da invocação do BLAST é o alinhamento da seqüência de entrada com as seqüências alvos do PDB. A invocação do próximo programa do workflow MHOline é o BATS e acontece de forma semelhante. Primeiramente é feita a cópia <assign> do resultado do BLAST para o container de entrada do BATS. Em seguida o BATS é invocado e o resultado já são os *scripts* de entrada para a geração dos modelos tridimensionais no Modeller. Com esses dados, a invocação do Modeller é feita

de forma semelhante pelo mesmo comando <invoke>, que executará o serviço Web correspondente e produzirá as coordenadas tridimensionais da seqüência de entrada. Por fim, este resultado é retornado para o usuário que iniciou o workflow MHOLline através do comando <reply> e do <container name="response">, podendo ser analisado ou até mesmo visualizado em qualquer programa de visualização de estruturas 3D como o RasMol [127] (Figura 39). A execução de todo esse processo é de responsabilidade da máquina de workflow BPWS4J 1.0.1, feito pela IBM, onde se encontra instalada a definição do workflow.



Figura 39 - Visão tridimensional de uma Proteína

Para cada instância de execução do workflow é gerado um número identificador chamado de TransID, que serve como elo entre as diversas execuções de cada uma das aplicações. Esse número é gerado por um serviço Web.

Esse identificador serve basicamente para identificar os dados intermediários que são gerados a cada etapa do workflow permitindo facilmente a recuperação dos mesmos no futuro. Esse identificador é gerado baseado na data e hora do servidor de aplicação e é único para todas as execuções de workflows. Um exemplo de um número TransID é 20040630_025635.

Além disso, também é armazenado no servidor MySQL, para cada instância do workflow, informações a respeito de qual, quando e quem executou cada experimento no ambiente 10+C. Com essas informações é possível saber qual workflow (pelo identificador TransID) gerou quais resultados, e se necessário, re-executar um workflow específico, mudando algum parâmetro de forma a refinar a pesquisa científica.

Para fornecer a possibilidade de re-execuções parciais de workflows, foram utilizados os serviços Web de persistência para recuperar os dados desejados utilizando o identificador `TransID` como parâmetro. Dessa forma, foi especificado um workflow que possuía dois parâmetros adicionais de entrada. O próprio `TransID`, e um número indicando qual é o passo em que o workflow iniciará. Assim, se for indicado o passo 4 como início, o workflow não irá executar os serviços Web correspondentes aos passos 1, 2 e 3, mas sim, executar os serviços Web de persistência para recuperar os dados de uma execução passada através do identificados `TransID`. O resto do workflow, a partir do passo 4 até o seu final, seria executado normalmente.

O mais interessante desta proposta, é que o workflow também é descrito como um serviço Web (Figura 40), desfrutando de todos os benefícios que esta tecnologia oferece e ainda, podendo até mesmo ser utilizado como parte de outros workflows. No arquivo WSDL do workflow, existe apenas uma seção adicional chamada `<Service Link Type>` para indicar o papel que cada serviço possui quando interage com os outros.

```
<message name="request">
  <part name="in1" type="xsd:string"/>
  <part name="in2" type="xsd:string"/>
  <part name="in3" type="xsd:string"/>
  <part name="in4" type="xsd:string"/>
</message>
<message name="response">
  <part name="output" type="xsd:string"/>
</message>
<portType name="runnerPT">
  <operation name="runnerOP">
    <input message="tns:request"/>
    <output message="tns:response"/>
  </operation>
</portType>
```

Figura 40 - Arquivo WSDL do WorkFlow MHOLline

Todas as etapas deste workflow foram executadas de forma automática. Poderíamos ainda ter incluído neste processo outras atividades como: testes de valores, execução em paralelo e desvios no fluxo de execução, com a utilização dos comandos adicionais `<flow>`, `<pick>`, `<switch>` e outros do BPEL.

Avaliação do Ambiente 10+C

“- Ele está diminuindo o número de giros por segundo - explicou o Sr. Foster. O pseudo-sangue circula mais devagar; por conseguinte, passa pelos pulmões a intervalos mais longos; portanto fornece menos oxigênio ao embrião. Nada como a penúria de oxigênio para manter um embrião abaixo do normal. (...) Quanto mais baixa é a casta, menos oxigênio se dá. O primeiro órgão afetado era o cérebro. Em seguida, o esqueleto. Com setenta por cento de oxigênio normal, obtinham-se anões. (...) - Os quais não são de nenhuma utilidade - concluiu o Sr. Foster”.

Aldus Huxley. Admirável Mundo Novo

Este capítulo é dedicado à avaliação do Ambiente 10+C. Na seção 3.3, foram definidas dez características para a definição e execução de workflows com base na análise dos trabalhos relacionados e na experiência de interação com biólogos e seus workflows em *scripts*. Essas características estão divididas em três grupos: quanto às questões referentes à definição de workflows, quanto à execução e quanto ao ambiente de implementação e utilização. Na primeira seção deste capítulo, é visto como o Ambiente 10+C atende cada uma dessas dez características. Nas duas últimas seções do capítulo são discutidas as diversas vantagens que esta abordagem possui sobre a abordagem tradicional de linguagem *scripts*. Na seção 5.2 é feita uma avaliação em termos qualitativos entre as duas abordagens enquanto que na seção 0 o workflow MHOLline é utilizado para uma comparação numa situação real com o ambiente tradicional de *scripts* e assim, comprovar a viabilidade da utilização de workflows e serviços Web como alternativa aos experimentos *in silico*.

5.1 Atendimento às Características 10+C

Definição abstrata do workflow: a definição do workflow no ambiente 10+C é feito através da linguagem BPEL, que é uma linguagem específica para a definição de processos. Através de construtores especializados em um arquivo semi-estruturado XML, eleva-se a modelagem do workflow para um nível de abstração mais alto, o que fica mais evidente quando se utilizam ferramentas gráficas nesta definição. A definição em BPEL em si, não está presa a nenhuma implementação específica de serviços Web. Apenas no momento da instalação (*deploy*) deste arquivo na máquina de execução de workflows será necessário informar o serviço Web a ser utilizado.

O Ambiente 10+C tem os recursos necessários para a definição de workflows sequenciais a partir dos serviços Web registrados. Além disso, podem-se utilizar ferramentas gráficas disponíveis no mercado e na comunidade para construir workflows mais complexos e utilizando outros serviços Web.

Definição do workflow em nível de programas executáveis: É necessário, de alguma maneira, mapear a definição de mais alto nível, para o nível de programas executáveis. Isto é feito no Ambiente 10+C através da utilização de aplicações científicas de serviços Web na execução de workflows. Dessa forma, não existe apenas uma especificação abstrata de workflow, mas sim uma especificação executável e que produz resultados concretos do experimento em estudo.

Geração automática do workflow: Uma vez definido o workflow utilizando o Ambiente 10+C, o próprio ambiente possui os recursos necessários para gerar, automaticamente, os arquivos BPEL e WSDL que possuem a especificação do workflow e a descrição da interface, respectivamente. Esses arquivos serão posteriormente instalados na máquina de workflow para a execução dos experimentos.

Execução do workflow: o ambiente 10+C fornece a execução dos workflows escritos em BPEL através do mecanismo de execução da IBM, o BPWS4J.

Re-execuções totais e parciais do workflow: Essa funcionalidade é fornecida no Ambiente 10+C através do armazenamento dos dados intermediários das execuções dos workflows, dos serviços Web específicos para a recuperação desses dados e pela utilização de um número identificador para cada instância de workflow chamado de

TransID. Na execução do workflow pelo Ambiente 10+C é indicado o número do passo em que se deseja que o workflow inicie e os passos anteriores serão recuperados pelo identificador TransID ao invés de serem re-executados.

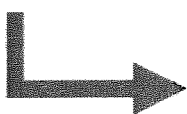
Tratamento de exceções na execução do workflow: a linguagem de definição BPEL possui suporte ao tratamento de erros e execução de atividades compensatórias através do uso dos construtores <correlations> e <compensationHandler>. Caso este tipo de tratamento seja necessário, deverá ser indicado que uma determinada ação no workflow, como a execução do BLAST, possui uma atividade compensatória relacionada. Essa indicação é feita através do <correlations>. Caso haja algum problema na execução do BLAST, a correlação é acionada e o trecho definido através de <compensationHandler> é executado. A atividade compensatória pode ser a execução de um BLAST de outra máquina, por exemplo.

Registro da execução de programas: A cada execução de um serviço Web pelo workflow, o serviço Web de persistência de dados é acionado para armazenar os dados intermediários de cada execução. Como visto na seção 4.5, os dados são armazenados na própria estrutura de diretórios do servidor de aplicação, podendo ser visualizados e utilizados em novas execuções posteriormente.

Registro da execução de workflows: Cada instância de workflow também tem sua execução registrada no Ambiente 10+C. Os dados referentes a qual workflow foi executado, qual o número identificador atribuído, quem o executou e quando isso aconteceu foram registrados através de outro serviço Web que armazenava estes dados no banco de dados MySQL. Posteriormente, pelo portal do Ambiente 10+C, existe a possibilidade de visualizar o registro de todas as execuções de workflows como pode ser visto na Figura 41.

Registro de Execução de Workflows

Workflow ID	Data	Hora	Usuário
MHOLLine 20040904_020000	2004-09-20	02:00:00	targino
MHOLLine 20040904_123045	2004-09-20	12:30:00	targino



Registro de Execução de Programas do Workflow

```
Workflow MHOLLine
BLAST
  blast 20040904_123045 IN
  blast 20040904_123045 OUT
BATS
  bats 20040904_123045 IN
  bats 20040904_123045 OUT
Modeller
  mod 20040904_123045 1 IN
  mod 20040904_123045 2 IN
  mod 20040904_123045 OUT
```

Figura 41 - Registro de execução de workflows

Execução remota de programas utilizando padrões abertos: o ambiente 10+C utiliza programas através do padrão de serviços Web não existindo nenhum tipo de interface proprietária envolvida em qualquer etapa do ciclo de vida de um experimento.

Distribuição através da Internet: o portal do Ambiente 10+C é acessado através de páginas HTML na Internet. Além disso, todos os programas científicos utilizados no Ambiente 10+C estão disponíveis através de serviços Web e os que não estão podem ser disponibilizados através da construção de tradutores, como visto na seção 4.5. Além disso, o próprio workflow construído na linguagem BPEL e instalado no servidor de aplicação, também é disponibilizado como um serviço Web, permitindo que todo o ambiente seja acessado e executado através da Internet utilizando padrões abertos.

5.2 Avaliação Qualitativa

Nesta seção, apresentamos a comparação da implementação do workflow MHOLline entre a abordagem tradicional que utiliza as linguagens de *scripts* e a nossa abordagem que utiliza workflows científicos através de serviços Web.

O Caos dos *Scripts*

Atualmente, existe uma grande falta de suporte para a definição, execução e registro de um experimento científico. Os biólogos e cientistas tendem a procurar soluções caseiras, isto é, soluções em algum ambiente que eles conhecem bem, e aspectos como simplicidade, legibilidade, manutenções e evoluções acabam sendo sacrificados.

Para a definição do experimento, os biólogos normalmente utilizam alguma linguagem de programação ou linguagem de *script*, ou no pior caso, seguem apenas o roteiro que possuem em sua própria cabeça para a execução dos programas na sequência correta. Neste caso, cada um dos programas é chamado manualmente, pegando a saída de um deles, renomeando-a e formatando-a para transformá-la na entrada de dados do próximo programa. Os arquivos de entrada e saída são normalmente organizados em uma estrutura de diretórios no sistema operacional e nomeados manualmente, muitas vezes sem nenhuma uniformidade, com o título do experimento em questão, o que não é muito confiável e difícil de reutilizar. Conforme o aumento do número de programas,

esse problema se agrava ainda mais, tornando praticamente impossível a composição das cadeias de programas.

A utilização de linguagens de programação ou *scripts* (como Perl, Python ou Java), possui o benefício de automatizar o processo de chamada de programas, passagem de dados e parâmetros e até mesmo a execução de *parsers* e formatação sobre os dados. Porém, como essas linguagens não são específicas para a composição de programas elas escondem muita informação sobre o experimento na complexidade de um código fonte. Os programas envolvidos, os dados de entrada e saída e o próprio fluxo de execução do workflow não ficam transparentes e são de difícil visualização. Para piorar, muitas dessas informações normalmente estão “*hard coded*” no código, gerando grande parte dos problemas citados anteriormente: dificuldade de entendimento por outras pessoas, dificuldade de evolução e dificuldade de manutenção. Isto é, pode ser fácil usar *scripts*, mas é muito difícil de reutilizá-los.

Para piorar, tarefas comuns são muitas vezes replicadas nos diversos centros de pesquisa pelo mundo, cada um fazendo o seu próprio *script* com a sua própria linguagem. Além disso, cada tarefa própria (ou *script* próprio) de um centro tem a indesejável característica de não se comunicar e não trabalhar junto com outros centros. (um *parser* de um centro não funciona com o conversor de dados do outro, etc.) O resultado de tudo isso, é que quando ocorre uma mudança evolutiva em algum programa ou base de dados, milhares e milhares de *scripts* em diversos centros de pesquisa precisam ser reescritos para voltarem a funcionar corretamente, em um imenso re-trabalho mundial.

Essa abordagem pode funcionar bem quando o *script* é utilizado por uma única pessoa executando um experimento em uma única máquina. Entretanto, sua utilização em um ambiente multi-usuário como a Internet, onde várias pessoas podem estar compartilhando definições de workflow e executando o mesmo experimento simultaneamente se mostra de difícil tratamento. Como os *scripts* trabalham com chamadas fixas a programas, existe uma enorme dificuldade para a execução de maneira transparente quanto à localização destes programas (local ou remoto) e até mesmo sobre direitos de acesso dos usuários a estes programas. Outra dificuldade na disponibilização dos workflows em forma de *scripts* se refere à portabilidade das linguagens utilizadas. A maioria dessas linguagens, até mesmo Java, não são 100% portáveis para qualquer

sistema operacional, gerando uma grande necessidade de alguma tecnologia que seja mais fácil de utilizar em qualquer tipo de ambiente.

Para finalizar, a visualização de resultados por usuários diferentes daqueles que executaram o workflow também apresentam dificuldades, pois os resultados parciais e finais podem estar disponíveis em uma estrutura local somente acessível ao usuário que executou o experimento.

Comparação do Experimento MHOLline

Nas linguagens de *script*, o MHOLline é construído em forma de código fonte Perl que contém as chamadas aos executáveis de cada programa participante (BATS, BLAST, Modeller, etc), como visto na Figura 42. Essas chamadas são feitas através do sistema de arquivos do sistema operacional indicando o caminho completo onde o programa reside [130]. No Ambiente 10+C, workflows e serviços Web residem em um servidor de aplicação na Internet e são invocados através de páginas Web.



Figura 42 – Sequência de execução do workflow MHOLline

No MHOLline *script*, a entrada de dados do workflow é feita indicando o caminho do arquivo FASTA. Esse arquivo é lido linha a linha no *script* (Figura 43-A) para posteriormente ser utilizado na execução do primeiro programa, o BLAST (Figura 43-B). Esse programa é invocado através de uma chamada ao sistema operacional e os arquivos de saída gerados são gravados em um diretório local para depois serem utilizados como entrada do próximo programa no fluxo de execução. Alguns desses arquivos precisam ser previamente manipulados de forma a extrair apenas a informação relevante (Figura 43-C). Desvios no workflows são implementados através das estruturas de desvios da linguagem de programação do *script* (Figura 43-D). Com isso, muitas vezes não é possível fazer um único *script* para representar um workflow. Vários *scripts* separados são gerados e não fica registrado que esses *scripts* compõem um workflow.

Esse modelo funciona bem quando o *script* é utilizado por uma única pessoa, em uma única máquina. Entretanto, sua utilização em um ambiente de produção onde

múltiplos usuários precisam executar e construir workflows, cada um com seus dados e parâmetros específicos, não se mostrou adequado. Esse tipo de situação é bem frequente no laboratório do IBCCF/UFRJ, transformando a construção, manutenção e evolução deste experimento em uma tarefa difícil e que consome muito tempo, com muita dependência do profissional de bioinformática.

Linguagens de *scripts* não são específicas para a composição de programas e por isso escondem muita informação na sua estrutura. A definição do experimento fica embutida implicitamente no *script* e existe uma grande dificuldade de entendimento para alguém que não conheça a linguagem Perl. Nomes de diretórios, arquivos e até mesmo parâmetros de entrada podem estar fixos no *script* dificultando manutenções e modificações no fluxo da execução. Além disso, apesar de ser uma linguagem interpretada, *scripts* não são totalmente portáteis entre diferentes sistemas operacionais.

Na abordagem de serviços Web, o workflow pode ser definido utilizando os construtores da linguagem BPEL. Cada programa utilizado, transformado agora em um serviço Web, é invocado apenas de uma única maneira, utilizando o construtor `<invoke>` (Figura 43-F). Passagens de dados entre um serviço Web e outro é feita explicitamente na definição do workflow utilizando estruturas de dados e construtores específicos (Figura 43-E). Testes e desvios na execução são igualmente, facilmente tratados (Figura 43-G).

<pre> # Open FASTA File my @file = common->read file(" \$s->{fastafilename}"); foreach my \$x (@file) { \$file .= "\$x"; } ... # Execute Blast \$blastdir = "/home/prgs/blast"; \$blastdir/blastall -i \$blastin -d \$blastdir -p blastp -o \$blastout ... # Parser file my %fields; read(STDIN,my \$temp,\$ENV{'CONTENT_LENGTH'}); my @pairs=split(/&/,-,\$temp); foreach my \$item(@pairs) { (my \$key, my \$content)=split(/=/,\$item,2); \$content=~tr/+//; \$content=~s/%{...}/pack("c",hex(\$1))/ ge; \$content=~s/\\t/ /g; \$fields{\$key}=\$content; } ... # Test flow if (\$score >= 100 { # Execute Modeller } ... </pre>	<pre> <sequence name="sequence"> <receive name="receive" partner="caller" ... <assign name="requestToServiceblast"> <copy> <from container="request"... <to container="blastrequest" </assign> ... <invoke name="invoke" partner="providerblast" operation="runBlast" ... /> <switch name="check-blast"...> <case condition= "bpws:getContainerData ('blastout','score') >= 100"> partner="modeller" portType="lms:modellerPT" ... </case> <otherwise> <terminate> </otherwise> </switch> </pre>
---	--

Figura 43 - Comparação entre um script Perl e um workflow definido com BPEL

A representação do workflow MHOLine nessa linguagem de mais alto nível traz inúmeras vantagens sobre a abordagem anterior, facilitando o processo de especificação e modificação de um workflow. A simples utilização de um modelo com maior abstração, permite aos biólogos maior clareza na visualização do processo como um todo, não escondendo os detalhes do fluxo de execução na complexidade de um código fonte. Dessa forma, tarefas como adicionar e remover programas no workflow, elaborar processos alternativos, definir atividades para compensação de erros e detectar possíveis falhas no processo atual podem ser facilmente endereçadas.

Como é mais fácil para uma pessoa entender a definição do workflow, também é mais fácil para um grupo de pesquisadores discutirem e decidirem sobre um experimento. Uma especificação de mais alto nível serve inclusive para a modelagem de alto nível do experimento e o melhor entendimento entre biólogos. O mais interessante é que a modelagem de mais alto nível é a própria definição do workflow.

A utilização de serviços Web para aplicações científicas e persistência de dados permite uma maior flexibilidade e independência por parte dos experimentos. Primeiro, por que a definição de workflows não fica limitada a programas disponíveis no ambiente local do cientista ou dentro da sua rede. Ele pode usar qualquer aplicação publicada na Internet como um serviço Web. Segundo, por que esta mesma tecnologia permite também que um serviço Web seja facilmente substituído por outro, desde que eles tenham a mesma interface, definida no arquivo WSDL.

Outro benefício que pode surgir com a adoção da abordagem de serviços Web é a possibilidade de re-execuções parciais de workflows já executados, economizando tempo e recursos dos biólogos. Dessa forma, um workflow seria executado apenas em sua parte final, recuperando os dados resultantes da parte inicial através de mecanismos de recuperação de dados. Esses mecanismos de persistência de dados intermediários das execuções, viabilizados através de serviços Web, é que permitem o registro da execução de experimentos e aplicações científicas.

Além disso, poderiam também ser incluídos nessa abordagem mecanismos de autenticação, segurança e execução dinâmica de workflows. Dessa forma, vários programas equivalentes para a execução de uma atividade, como por exemplo, as várias versões do programa BLAST (NCBI, WU, etc), seriam escolhidos no momento da

execução, aquele que fosse mais apropriado, segundo algum critério de custo (desempenho, proximidade, custo monetário, etc.) [17][18].

Podemos resumir as inúmeras vantagens da abordagem de workflows científicos e serviços Web na Tabela 2.

Tabela 2 - Vantagens da abordagem de workflows científicos e serviços Web

Característica	Scripts	Workflows e Serviços Web
Definição do Experimento	<ul style="list-style-type: none"> • Feito na própria linguagem de <i>script</i>. • O fluxo de execução, os programas utilizados e os dados estão implicitamente embutidos na complexidade do código fonte. • Podem existir informações de modo rígido (<i>hard coded</i>) no código. 	<ul style="list-style-type: none"> • Melhor entendimento e facilidade na definição do experimento pela utilização de uma abstração de mais alto nível. • Isolamento explícito entre a definição do workflow e o processo de invocação e execução das aplicações. • Possibilidade de utilizar ferramentas gráficas para a definição do workflow com diagramas, grafos, setas e outros recursos visuais. • Melhor comunicação entre uma equipe de biólogos durante o processo de definição do experimento.
Manutenção e Evolução do Experimento	<ul style="list-style-type: none"> • É preciso conhecer a linguagem de <i>script</i> e depurar o código fonte. • Modificações nas aplicações científicas geram adaptações em todos os <i>scripts</i> que utilizam estes programas. 	<ul style="list-style-type: none"> • Maior facilidade de alterações e manutenções futuras pela utilização de uma definição abstrata e de ferramentas gráficas. • Modificações nas aplicações científicas geram adaptação em um único serviço Web.
Suporte a Laboratórios Virtuais de Bioinformática	<ul style="list-style-type: none"> • Dificuldade de compartilhamento de dados e experimentos, pois o ambiente é dependente da estrutura de diretórios e direitos de acesso do sistema operacional. • Proliferação excessiva de scripts em diversos centros de pesquisa que fazem a mesma tarefa. • Grande dificuldade de interoperabilidade entre as diversas aplicações científicas e os diversos experimentos. 	<ul style="list-style-type: none"> • Compartilhamento dos experimentos entre biólogos e centros de pesquisa através da Internet • Maior produtividade e economia do tempo dos biólogos pela utilização de experimentos científicos já prontos e disponíveis na comunidade através de serviços Web. • Existe um padrão para a interoperabilidade
Execução de Experimentos	<ul style="list-style-type: none"> • Funciona quando um <i>script</i> é utilizado por um único cientista, mas apresenta grande dificuldade para um ambiente onde o mesmo experimento é executado por várias pessoas. • A execução de diversas instâncias de um experimento é um processo 	<ul style="list-style-type: none"> • Os experimentos são compartilhados através da Internet • Maior facilidade na execução de múltiplas instâncias do experimento alterando apenas os valores dos parâmetros de entrada. • Possibilidade de re-execuções

	<p>lento e trabalhoso.</p> <ul style="list-style-type: none"> • Re-execuções de experimentos constituem em um processo igualmente lento e trabalhoso. • Resultados intermediários ficam disponíveis apenas a quem executou o experimento. • É fácil de o biólogo se perder em diversos dados resultantes, se o mesmo não for muito organizado • A execução do experimento fica presa a aplicações científicas escolhidas inicialmente. 	<p>parciais do experimento a partir de um determinado ponto já processado.</p> <ul style="list-style-type: none"> • Identificação da procedência e das transformações sobre os dados é suportada e acessada por todos. • Adiciona mais conhecimento e confiabilidade aos resultados dos experimentos e disponibiliza-os em um ambiente centralizado. • Independência do workflow entre as diversas implementações de uma determinada aplicação, já que pode-se alterar o workflow para utilizar outro serviço Web.
--	--	---

Conforme analisado em [44], apesar de todas as vantagens da tecnologia de serviços Web sobre as linguagens de *scripts*, muito ainda precisa ser feito no suporte de serviços Web a workflows. Isoladamente, os serviços Web não são suficientes para suportar por completo um ambiente de experimentos de workflows científicos. Embora seja poderoso para resolver questões de interoperabilidade e composição de diversos serviços, existe uma grande necessidade de adicionar um suporte maior nas descrições semânticas dos recursos científicos, de forma a permitir aos cientistas e biólogos procurar e encontrar os serviços que mais se adequam as suas necessidades. Neste contexto, a tecnologia de serviços Web deve ser complementada com outras tecnologias, como a adoção de metamodelos [42] ou ontologias [14][179], para construir um ambiente mais robusto para a composição de workflows científicos.

5.3 Avaliação Quantitativa

A próxima etapa do nosso trabalho consistiu em validar a arquitetura proposta, comparando os resultados obtidos e os tempos de execução do workflow do MHOLine em cada uma das duas abordagens: o ambiente tradicional de linguagens de *scripts* e a arquitetura de serviços Web. Dessa forma, elaborou-se um experimento para medir os tempos apenas da execução das duas infra-estruturas, desconsiderando, portanto, os tempos das execuções dos programas externos envolvidos (BLAST, BATS e Modeller), que seriam os mesmos em cada uma das abordagens. O workflow MHOLine foi executado dez vezes em cada ambiente tendo como entrada um arquivo FASTA com uma única seqüência de proteínas (Figura 44).

```

>gi|15640035|ref|NP_062587.1| thiophene and furan oxidation protein ThdF [Vibrio cholerae]
MALIQSCSGNTMTTDTIVAQATAPGRGGVGIIRVSGPLAAHVAQTVTGRTLPRRYAEYLPFTDEDGQQLDQGIALFFPNPHSFTGEDVLE
LQGHGGPVVMDMLIRRLQIKGVRPARPGEFSEAFNLNDKMDLTQAEAIADLIDASSEQAAKSALQSLQGEFASKRIHTLVESLIHLRIYV
EAAIDFPBEEIDFLADGKVSADLQTTIDNLA AVRREANQGAIMREGMKVVIAGRPNAGKSSLLNALS GKESAIVTDIAGTTRDVLREHIH
IDGMPLHIIDTAGLRDASDAVEKIGIERAWEEIRQADRVI.FMVDGTTTEATDPQDIWPDFVDKLPENIGITVI.RNKADQTGEPLGICHVN
QPTLIRLSAKTGQGV DALRQHLKECMGFSGNQEGGFMMARRRHLDALERAABHLAIGQQQLEGYMAGEIILAEELRIAQQHLNEITGEFSSD
DLLGRIFSSFCIGK

```

Figura 44 - Sequência FASTA de entrada do MHOLline

A média das dez execuções foi calculada. Todo o experimento foi executado em uma única máquina. O workflow em *script* Perl chamava cada um dos três programas externos diretamente pela estrutura de diretórios do sistema operacional. Por outro lado, tanto o workflow de serviços Web, como os serviços Web correspondentes a cada um dos programas, residiam em um servidor de aplicações e eram invocados por uma aplicação Java local na mesma máquina. A máquina utilizada foi um PC com processador AMD Athlon XP 1800 com 512Mb de memória. O resultado final (a estrutura molecular resultante) foi exatamente a mesma em ambas as abordagens.

O experimento consistiu da medição dos tempos do início da execução do workflow até o momento imediatamente anterior à execução do primeiro programa (Figura 45 – Passo 1), o BLAST. Depois, foi medido o tempo da segunda etapa que iniciou após o fim da execução do BLAST até próximo programa (Figura 45 – Passo 2) e assim sucessivamente até o final do workflow (Figura 45 – Passo 3 e 4). O objetivo dessa medição era capturar os custos de execução de cada uma das infra-estruturas. Nos *scripts* Perl, o custo era basicamente em chamar programas executáveis locais. No ambiente de serviços Web, esses passos eram mais complexos, havendo a necessidade de empacotar os dados e parâmetros de entradas em mensagens SOAP, enviar ao servidor de aplicação para a execução do serviço e desempacotar os resultados obtidos.

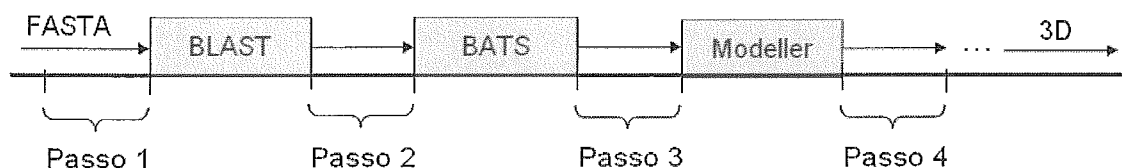


Figura 45 – Medição de tempo na execução do workflow MHOLline

Como já era esperado, a execução dos *scripts* apresentou-se muito mais rápida conforme pode ser visto na Tabela 3. Entretanto, apesar dos tempos da arquitetura de serviços Web estarem em média uma ordem de grandeza acima, esses tempos não podem ser considerados proibitivos. Na verdade, se compararmos estes tempos com os tempos da execução dos programas propriamente ditos e conseqüentemente com o

tempo total de execução do workflow, o custo da arquitetura de serviços Web seria totalmente desprezível. Para se ter uma idéia, o programa Modeller, o mais custoso, demorou em média 245 segundos para ser executado (Tabela 4). O tempo de 2,2 segundos da utilização serviços Web não constituiu nem 1% deste custo e pode ser em parte ignorado. Ainda sim, como o experimento foi feito inteiramente em uma única máquina, os custos de transmissão de rede não foram considerados, o que provavelmente diminuiria ainda mais a fatia da infra-estrutura dos serviços Web sobre o custo total de execução.

Tabela 3 - Tempo de execução das infra-estruturas de serviços Web e linguagens de script (em segundos)

Passos do MHOLline	Serviços Web	Script Perl
Passo 1 – Início MHOLline → BLAST	0.942	0.023
Passo 2 – BLAST → BATS	0.764	0.016
Passo 3 – BATS → MODELLER	0.330	0.026
Passo 4 – MODELLER → Fim MHOLline	0.209	0.024
TOTAL	2.245	0.089

Desse modo, consideramos que a abordagem de serviços Web é totalmente viável de ser implementada em um ambiente de produção, e que o aparente tempo adicional deste caminho seria inteiramente compensado pelos benefícios atingidos citados anteriormente: maior flexibilidade, clareza, interoperabilidade e produtividade. Esses benefícios justificariam inclusive, se este tempo adicional aumentasse se fossem utilizadas múltiplas seqüências ou grandes seqüências como entrada para o workflow.

Tabela 4 - Tempo de execução dos programas de bioinformática (em segundos)

Programas	Tempo de Execução
BLAST	0.161
BATS	0.095
MODELLER	245.257

“As crianças Alfas vestem roupas cinzentas. Elas trabalham muito mais do que nós porque são formidavelmente inteligentes. Francamente, estou contentíssimo de ser um Beta, porque não trabalho tanto. E além disso, somos muito superiores aos Gamas e aos Deltas. Os Gamas são brancos. Eles se vestem de verde e as crianças Deltas se vestem de cáqui. Oh, não, não quero brincar com crianças Deltas. E os Ipsilons são ainda piores. São demasiado brancos para saberem ler e escrever. E além disso, se vestem de preto, que é uma cor horrível. Como sou feliz por ser um Beta. (.!). Eles ouvirão isso cento e vinte vezes, três vezes por semana, durante trinta meses. Depois disso, passarão a uma lição mais adiantada”.

Aldus Huxley. Admirável Mundo Novo

O crescimento exponencial da utilização de recursos computacionais foi um dos fatores que consolidaram os grandes avanços da área de bioinformática, como por exemplo, a pesquisa do genoma humano. Por outro lado, a necessidade de métodos e processos para automatizar, gerenciar e apoiar o trabalho diário de cientistas e biólogos acompanhou este ritmo de crescimento e se tornou um grande desafio a ser resolvido [81][100][133].

Dentre as inúmeras subáreas de pesquisa, o auxílio aos cientistas no processo de experimentação científica tem se destacado como um dos principais objetivos nos dias atuais. Cada vez mais, etapas dos experimentos que eram feitos em laboratórios, estão sendo complementadas ou em alguns casos substituídas por atividades totalmente executadas e analisadas através do uso de computadores, no que se convencionou chamar de experimentos *in silico*. A grande parte desses experimentos consiste na composição de diversos programas em uma cadeia de execução, onde a saída de um deles serve como entrada de dados do próximo. O conjunto de diversas execuções nesta cadeia de programas, mudando os dados de entradas e calibrando parâmetros, é o que

fornece subsídios aos cientistas para efetuarem o ciclo de experimentação e teste, inerente ao processo de busca de novos conhecimentos.

Entretanto, mesmo o cientista mais organizado encontra grande dificuldade em gerenciar a complexidade que se tornou o uso de aplicações científicas de bioinformática. Linguagens de *scripts*, como Perl [124], encontraram enorme aceitação na comunidade para auxiliar o dia a dia dos cientistas, devido a sua grande facilidade de implementação e o extenso suporte ao tratamento de arquivos texto. Apesar de parecerem uma boa solução, essas linguagens escondem uma grande armadilha por trás do seu uso. Linguagens de *script* não são boas para descrever um experimento científico, pois embutem a semântica do mesmo na complexidade de um código fonte. Modificações e manutenções também se constituem em um processo altamente trabalhoso. Além disso, na execução deste experimento, existe uma grande rigidez nos programas que poderão ser utilizados, nos acessos simultâneos de usuários e no compartilhamento de resultados.

Esta dissertação propôs uma nova forma para tratar os experimentos de bioinformática através da utilização de workflows científicos [2] e serviços Web [160]. Os workflows foram utilizados para a definição e execução dos experimentos. Os serviços Web foram utilizados para resolver o problema da interoperabilidade das diversas aplicações científicas de forma a torná-las disponíveis a qualquer experimento. Um portal Web foi construído para dar suporte aos cientistas nessas duas tarefas.

Este ambiente de definição e execução de experimentos *in silico* foi chamado de Ambiente 10+C, pois satisfaz dez características significantes a este tipo de problema. Genericamente, podemos resumir essas características em três conceitos principais: definição do experimento em mais alto nível, execução do experimento com suporte a re-execuções e registro, e, por último, utilização de infra-estrutura de código aberto através da Internet.

A definição de experimentos em mais alto nível foi resolvida através da utilização de workflows científicos. Através da utilização da linguagem BPEL [52], elevou-se para um nível maior de abstração o processo de descrição do experimento, trazendo maior entendimento e facilidade para os cientistas, melhorando a comunicação entre eles e facilitando futuras modificações e evoluções, já que a descrição do experimento fica

claramente separada do fluxo de execução. O próprio portal do Ambiente 10+C pode ser utilizado para a definição dos experimentos científicos através de uma interface visual que guia o biólogo para a escolha do fluxo de execução do experimento. Ao final, o arquivo BPEL é automaticamente gerado ficando pronto para ser utilizado.

A execução do experimento foi feita através da interpretação de sua especificação pela máquina do workflow e a conseqüente invocação de cada programa envolvido. Serviços Web foram utilizados como base para permitir a total interoperabilidade entre os programas e os workflows. Além disso, adicionamos mecanismos para suportar o registro das execuções de programas e workflows, e a possibilidade de re-execuções de trechos parciais do experimento. A primeira característica é essencial para a identificação da procedência e das transformações executadas sobre os dados. A percepção sobre de onde veio o dado, quem o modificou e como ele chegou ao banco de dados é de suma importância para a confiabilidade que um cientista irá atribuir a ele. Com a utilização de workflows, a tarefa de armazenar este tipo de meta-informação, também conhecido como *data provenance* [34][35] ficaria automatizada e permitiria assim a possibilidade de descoberta de conhecimento adicional sobre esses dados. Técnicas de mineração de dados poderiam ser utilizadas para potencializar os resultados. O registro das execuções, através da persistência dos dados intermediários, também permite a segunda característica: a re-execução de apenas trechos de um experimento, aproveitando os resultados já obtidos até um determinado ponto, mas modificando parâmetros e entradas a partir dali. A execução propriamente dita foi feita através de uma máquina de execução de workflows disponível pela IBM, chamada BPWS4J [87].

A infra-estrutura de código aberto pela Internet é um fator determinante para a aceitação de qualquer ambiente pela comunidade científica. O portal do Ambiente 10+C foi construído para prover uma maior facilidade ao processo de descrição e execução de experimentos, centralizando as principais operações dos cientistas em um único lugar. O portal foi construído utilizando a linguagem Java [91], através de JSP e *Servlets*, e do banco de dados MySQL [111]. Através do Ambiente 10+C, pretende-se capturar as necessidades dos biólogos e cientistas com apenas alguns cliques do *mouse* e um mínimo de dificuldade.

A tecnologia de serviços Web é um padrão aberto já largamente adotado pela indústria e com grandes perspectivas de cada vez mais aumentar a sua base de utilização. Cada vez mais, muitos projetos de bioinformática também estão adotando a tecnologia de serviços Web [37][47][89][127][171][175] para tratar seus problemas de interoperabilidade e flexibilidade entre os diversos programas existentes nas diferentes plataformas. Em [129], os autores estão usando os serviços Web para agrupar e encontrar programas biológicos similares, mas sem considerar a sua composição. O projeto BioMoby [175] usa os serviços Web para publicar e encontrar dados e programas em um ambiente de código aberto, mas também sem descrever a composição e execução de workflows. Além disso, nenhum desses projetos mostrou a utilização de seus trabalhos em conjunto com experimentos reais de bioinformática.

A solução de workflows científicos para ambientes de experimentos em bioinformática também não é uma novidade na comunidade. A arquitetura SRMW [43] possui suporte à descrição de programas e workflows combinados com a gerência de recursos científicos, fortemente apoiado no uso de metamodelos [42], para adicionar maior semântica a esses recursos. A descrição de maior semântica facilita o processo de escolha de quais programas e quais parâmetros utilizar para um determinado experimento. Porém, não fez parte do escopo desta tese [40] a definição e à execução de workflows no nível operacional.

Projetos como LabBase/LabFlow [73][74], Chimera [66] e gRNA [98] se concentram na gerência de workflows, possuindo mecanismos para a definição e execução dos mesmos. Nenhum deles, porém, utiliza a tecnologia de serviços Web para a descrição e invocação de aplicações científicas, limitando-as em alguma infraestrutura fechada e proprietária.

O projeto MyGrid [179] tem basicamente as mesmas características dos projetos citados no parágrafo anterior, utiliza a infra-estrutura de serviços Web e Grids Computacionais, e também possui um amplo suporte à descrição semântica de dados e programas através de ontologias. Entretanto, assim como os anteriores, não houve exemplos maiores de sua utilização em experimentos reais de bioinformática.

Uma das principais contribuições deste trabalho foi exatamente o fato de estarmos trabalhando junto aos biólogos e utilizarmos um experimento real de bioinformática. O

experimento MHOLline, projetado por Rössle [130], para a predição de estruturas tridimensionais de proteínas foi implementado no Ambiente 10+C. Dessa forma não apenas sugerimos um ambiente para definição e execução de workflows, como também o implementamos em um caso real e mostramos como fazer isso [41][44].

Na verdade, o ambiente que construímos poderia ser utilizado para qualquer outro tipo de aplicação científica não ficando restrito ao ambiente de bioinformática, e essa poderia ser considerada outra contribuição importante desta dissertação. Já que a base do ambiente são os serviços Web, e este, como já falamos, possuem descrições e interfaces bem definidas (WSDL [161]), facilidade de comunicação e invocação (através de mensagens SOAP [162]) e facilidade de busca (através de registros UDDI [154]), poderíamos definir e executar workflows para qualquer serviço Web disponível na Internet.

Mesmo que alguma aplicação científica não estivesse publicada como um serviço Web, mas seja extremamente necessária para um determinado workflow, mostramos como seria o processo para disponibilizá-la nesta tecnologia, o que consideramos como mais uma contribuição desta dissertação.

Outra contribuição desta dissertação foi a definição de 10 características relevantes para ambientes de experimentos *in silico* utilizando workflows científicos. Em conjunto com os biólogos, capturando suas experiências pessoais e suas necessidades no processo de experimentação científica, elaboramos a lista de 10 características, chamadas de características 10+C, que serviu de base para a especificação no nosso ambiente e a posterior comparação com os outros trabalhos relacionados.

Uma das principais contribuições desta dissertação foi à comparação entre os ambientes de linguagens *scripts* e o ambiente de workflows científicos de serviços Web. Apesar de ser quase lugar comum na comunidade a crítica em relação ao primeiro, e que ambientes com maior facilidade de interoperabilidade e flexibilidade serem necessários, existe uma grande carência de comparações entre os dois. Nesta dissertação foi feita uma análise qualitativa entre esses ambientes, destacando as deficiências e vantagens de cada um, e por fim, foi executado um experimento real comparando tempos e resultados entre eles. A experiência mostrou que apesar da maior rapidez das linguagens *scripts*, a

diferença de tempo entre a abordagem de workflows pode ser ignorada, já que esse tempo é, em média, duas ordens de grandeza inferior ao tempo de execução dos próprios programas.

Acreditamos que a ampliação e utilização do Ambiente 10+C em um ambiente real de produção, possa beneficiar os biólogos a conduzirem os seus experimentos e melhorar o processo de experimentação em si. Experiências entre os biólogos poderiam ser mais facilmente trocadas e compartilhadas entre eles. Modificações em workflows já existentes seriam naturalmente tratadas. Cientistas e biólogos menos experientes poderiam aproveitar melhor os experimentos já existentes e aprimorar as suas análises.

A infra-estrutura do Ambiente 10+C é relativamente simples. Somente é necessário um servidor de aplicação para hospedar o portal, o banco de dados e a máquina do workflow. As aplicações científicas seriam disponibilizadas como serviços Web por toda a Internet e seriam apenas invocadas a partir do servidor. Caso uma aplicação científica não tenha sido disponibilizada, pode-se construir uma classe Java tradutora responsável por isso, como fizemos para os programas do experimento MHOLline.

Assim como o processo de busca de novos conhecimentos, as possibilidades de evoluções desta dissertação também são muitas. Em primeiro lugar, poderíamos aprimorar as descrições de programas e workflows através do uso de ontologias [77][78]. Dessa forma, ontologias genéricas ou ontologias mais específicas, como *Gene Ontology* [14], poderiam ser utilizadas para em conjunto com os metamodelos [137] da arquitetura SRMW [42] prover uma total abstração semântica para a classificação e busca dos serviços disponíveis.

Outra linha que poderia ser explorada, seria a inclusão de mecanismos para paralelizar as execuções do workflow e aumentar o desempenho de experimentos mais custosos [105][106]. Dessa forma, ao invés de um programa ficar esperando o término da execução do seu antecessor, o mecanismo de execução ficaria responsável por pegar os resultados parciais já obtidos e distribuí-los para os próximos programas, diminuindo o tempo de execução do workflow.

Nesta mesma linha, a elaboração de planos de execuções mais complexos, que avaliem os custos de diversos programas equivalentes (BLAST NCBI [31] ou WU-

BLAST [168], por exemplo) seguindo algum critério de qualidade [120] (maior desempenho, proximidade, custo monetário, etc.) poderiam ser embutidos na própria descrição do serviço, como extensões da linguagem WSDL, e utilizados pela máquina do workflow para a tomada das decisões. O trabalho de Azevedo *et al.* [17][18] propõe algumas dessas extensões para serviços Web mas não é específico para bioinformática. O trabalho de Lemos [102] também tem essas características, mas utilizou um ambiente fechado e proprietário ao invés do padrão de serviços Web. A utilização da plataforma de serviços Grid (*Grid Services*) também poderia ser considerada para transformar os serviços Web em objetos de software mais dinâmicos que possuem uma natureza transiente (criação e destruição do serviço) e armazenariam os atributos comentados anteriormente [101].

A integração do Ambiente 10+C com as aplicações de “*Electronic Notebook*” [148] também poderiam ser exploradas para promover uma interação automática entre o experimento científico e o seu caderno de anotações. Essas anotações são bastante utilizadas em experimentos científicos, principalmente na área farmacêutica, para registrar todo o histórico do experimento como textos, imagens, gráficos, histórico de testes e resultados atingidos.

Por fim, a construção do Ambiente 10+C teve seus objetivos atingidos pois conseguimos especificar, implementar e avaliar um solução para um experimento real de bioinformática. Apesar de ter implementado o experimento MHOLline como exemplo, qualquer outro experimento científico também poderia ser incorporado, já que o ambiente é genérico o suficiente para isso. Mostramos que a arquitetura de serviços Web é uma ótima opção para este tipo de ambiente e que workflows podem e devem ser explorados para a definição e execução de experimentos.

Referências Bibliográficas

- [1] Aalst, W., Hee, K. *Workflow Management: Models, Methods, and Systems*. MIT Press, January 2002.
- [2] Aalst, W., Hofstede, A., Kiepuszewski, B., Barros, A. "Workflow Patterns", *Distributed and Parallel Databases*, Vol. 14, num 3, pp. 5-51, 2003.
- [3] AceDB Database, Disponível em: <http://genome.cornell.edu/acedoc/index.html>, 2002.
- [4] Auerbach A. D. "8th International HUGO-Mutation Database Initiative Meeting, April 9, 2000, Vancouver, Canada". *Human Mutation*, v. 16, n. 3, pp. 265-268, September 2000.
- [5] Ailamaki, A. Ioannidis, Y. Livny, M. "Scientific Workflow Management by Database Management". In *Proceedings of 10th International Conference on Scientific and Statistical Database Management*, pp. 190-199, Capri, Italy, July 1998.
- [6] Aleixo, A. C. "Clonaram Gente". *Istoé On-Line*, 05/04/2002. Disponível em <http://www.terra.com.br/istoe/artigos/clone.htm>, 2004.
- [7] Altova - XML Development, Data Mapping, and Content Authoring Tools. Disponível em <http://www.altova.com/>, 2004.
- [8] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., Lipman, D. J. "Basic local alignment search tool". *Journal of Molecular Biology*, v. 215, pp.403-410, 1990.
- [9] Amabis, J. M., Martho. G. R. *Fundamentos da Biologia Molecular*. Editora Moderna, 1ª edição. São Paulo, 1990.
- [10] Ankolekar, A., Burstein, M., Hobbs, J., Lassila, O., Martin, D., McIlraith, S., Narayanan, S., Paolucci, M., Payne, T., Sycara, K., Zeng, H. "DAML-S: Semantic Markup for Web Services". In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, July 30-August 1, 2001.
- [11] Apache HTTP Server Project. Disponível em <http://httpd.apache.org/>, 2002.
- [12] Apache Tomcat. Disponível em <http://jakarta.apache.org/>, 2002.
- [13] Arkin, A., "Business Process Modeling Language (BPML)". Disponível em <http://www.bpml.org/bpml-spec.esp>, Março 2001.
- [14] Ashburner, M., Lewis, S. "On Ontologies for Biologists: the Gene Ontology - Uncoupling the Web", in *Proc. Silico Biology* 247, Novartis Found Symposium, pp. 66-83, 2002.
- [15] Axis Project. Disponível em <http://ws.apache.org/axis/>.
- [16] Azevedo, A. L., Braga, I. "Nasce no Brasil Bezerra Clonada de um Clone". *Jornal O Globo*, 20 de Fevereiro de 2004. Disponível em <http://www.jornaldaciencia.org.br/Detalhe.jsp?id=16466>.

- [17] Azevedo, V. "WebTransact-EM: Um Modelo para a Execução Dinâmica de Serviços Web". *Dissertação de M.Sc*, COPPE/Sistemas, UFRJ, Brasil, 2003.
- [18] Azevedo, V., Pires, P., Mattoso, M. "WebTransact-EM: Um Modelo para a Execução Dinâmica de Serviços Web Semanticamente Equivalentes" *In Proc. Simpósio Brasileiro de Sistemas Multimídia e WEB*, pp. 139-146, Salvador, 2003.
- [19] Bairoch, A., Apweiler R. "The SWISS-PROT protein sequence data bank and its supplement TrEMBL". *Nucleic Acids Research*, v. 26, n. 1, pp. 38-42, 1998.
- [20] Bairoch A. "The ENZYME database in 2000". *Nucleic Acids Research*, v. 28, n. 1, pp. 304-305, 2000.
- [21] Baker, D., Sali, A. "Protein Structure Prediction and Structural Genomics". *Science Magazine*, v. 294, pp. 93-96, October 2001.
- [22] Barclay T., Gray, J., Strand E., Ekblad S., Richter J. "TerraService.NET: An Introduction to Web Services". *Technical Report MS-TR-2002-53 Microsoft Research Advanced Technology Division*, June 2002.
- [23] Baxevanis A. "The Molecular Biology Database Collection: 2003 updated". *Nucleic Acids Research*, v. 31, n. 1, pp. 304-306, 2003.
- [24] BBC Brasil.com. "Empresa mostra no Brasil foto de suposto bebê clonado". BBC Brasil.com., 25 de março, 2003. Disponível em http://www.bbc.co.uk/portuguese/ciencia/030325_clonedi.shtml, 2004.
- [25] Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., Wheeler, D. L. "GenBank: update". *Nucleic Acids Research*, v. 32, n. 1, pp. 23-26, Jan 2004.
- [26] Berman, H. M., Battistuz, T., Bhat, T. N., *et al.* (2002). "The Protein Data Bank." *Biological Crystallography*, v. 58, n. 1, pp. 899-907, 2002.
- [27] Berman, H., Westbrook, J., Feng Z., *et al.* "The Protein Data Bank". *Nucleic Acids Research*, v. 28, n. 1, pp. 235-242, 2000.
- [28] Bernauer, M., Kappel, G., Kramler, G., Retschitzegger, W. "Specification of Interorganizational Workflows - A Comparison of Approaches". *In Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2003)*, pp. 30-36, 27-30 July, 2003, Orlando, USA.
- [29] Bhowmick, S. S., Cruz, P., Laud, A. "XomatiQ: Living with Genomes, Proteomes, Relations and a Little Bit of XML". *In Proceedings of ICDE*, pp. 857-868, Bangalore, India, 2003.
- [30] Bhowmick, S. S., Vedagiri, V., Laud, A. "HyperThesis: The gRNA Spell on the Curse of Bioinformatics Applications Integration, *in Proceedings of ACM International Conference on Information and Knowledge Management (CIKM 2003)*, pp. 402-409, New Orleans, 2003.
- [31] BLAST NCBI Laboratory. Disponível em: <http://www.ncbi.nlm.nih.gov/BLAST>, 2002.
- [32] BPWS4J - IBM. Disponível em <http://www.alphaworks.ibm.com/tech/bpws4j>.
- [33] Brittenham. P., Curbera, F., Ehnebuske, D., Graham, S. "Understanding WSDL in a UDDI Registry". IBM Software Group. Disponível em <http://www-106.ibm.com/developerworks/webservices/library/ws-wsdl/>, 2002.

- [34] Buneman, P., Khanna, S., Tan, W. "Data provenance: Some Basic Issues". *In Foundations of Software Technology and Theoretical Computer Science*, v. 1974, pp. 87-93, 20th Conference, (FST TCS) New Delhi, India, 2000.
- [35] Buneman, P., Khanna, S., Tan, W. "Why and Where: A Characterization of Data Provenance". *In Proceedings of the 8th International Conference on Database Theory*. v. 1973, pp. 316-330, 2001.
- [36] Burns, G. W., Bottino, P. J. *Genética*. Guanabara Koogan, 6ª Edição. Rio de Janeiro, 1991.
- [37] caBIO - The cancer Bioinformatics Infrastructure Objects, Disponível em <http://ncicb.nci.nih.gov/core/caBIO>, 2004.
- [38] CapeScience, Web Services Developer Community. "GlobalWeather". Disponível em <http://www.capescience.com/webservices/globalweather/index.shtml>, 2004.
- [39] CASP - Critical Assessment Structure Prediction. Disponível em: <http://www.ncbi.nlm.nih.gov/Structure/RESEARCH/casp3/index.html>.
- [40] Cavalcanti, M. "Scientific Resources Management: Towards an In Silico Laboratory", *Ph.D. Thesis*, Technical Report ES-605/03, COPPE/UFRJ, Brazil, 2003.
- [41] Cavalcanti, M., Baiao, F., Rössle, S., Bisch, P. M., Targino, R., Pires, P. F., Campos, M. L., Mattoso, M. "Structural Genomic Workflows Supported by Web Services". *In Proceedings of Workshop on Biological Data Management DEXA '03*, Prague, Czech Republic, pp. 45-49, 2003.
- [42] Cavalcanti, M., Mattoso, M., Campos, M. L., Llibat, F., Simon, E. "Sharing Scientific Models in Environment Applications". *In Proceedings of ACM Symposium on Applied Computing*, Madrid, pp. 453-457, 2002.
- [43] Cavalcanti, M., Mattoso, M., Campos, M. L., Simon, E., Llibat, F. "An Architecture for Managing Distributed Scientific Resources" *In Proceedings Int. Conf. on Scientific and Statistical Database Management*, Edinburgh, pp. 47-55, 2002
- [44] Cavalcanti, M., Targino, R., Baiao, F., Rössle, S., Bisch, P. M., Pires, P. F., Campos, M. L., Mattoso, M. "Managing Structural Genomic Workflows using Web Services". *Data and Knowledge Engineering Journal (DKE)*. Elsevier Science Publication, 2004. Available via Science Direct.
- [45] Chen, J. Y. "A Bioinformatics Discovery-oriented Computing Framework". PhD Thesis, University of Minnesota, Minneapolis, USA, 2001.
- [46] Chicurel, M. "Bioinformatics: Bringing it all Together". *Nature*, v. 419, n. 17, pp. 751-755, October 2002.
- [47] Clark, T. "Identity and interoperability in Bioinformatics". *Briefings in Informatics*, v. 4, n. 1, pp. 4-6, 2003.
- [48] Collins, Francis. S., Green, Eric D. Guttmacher, Alan. E. Guyer, Mark S. "A Vision for the Future of Genomics Research". *Nature*, v. 442, n. 24, pp. 835-847, 2003.
- [49] Condor-G. Disponível em: <http://www.cs.wisc.edu/condor/condorg/>.

- [50] Cornell, M., Paton, N.W., Wu, S., Goble, C.A., Miller, C.J., Kirby, P., Eilbeck, K., Brass, A., Hayes, A., Oliver, S.G. "GIMS - A Data Warehouse for Storage and Analysis of Genome Sequence and Functional Data". *In Proceedings of 2nd IEEE International Symposium on Bioinformatics and Bioengineering (BIBE)*, IEEE Press, pp.15-22, 2001.
- [51] Critchlow, T., Musick, R., Slezak T. "Experiences Applying Meta-Data to Bioinformatics". *Information Sciences Journal*, v. 139, n. 1-2, pp. 3-17, November 2001.
- [52] Curbera, F., Goland, Y., Andrews, T., *et. al*, "Business Process Execution Language for Web Services v1.1". Microsoft, BEA, IBM, May-2003. Disponível em: <http://www.ibm.com/developerworks/library/ws-bpel/>.
- [53] DagMan. Disponível em: <http://www.cs.wisc.edu/condor/dagman/>.
- [54] DAS Project. Disponível em: <http://www.biodas.org/>, 2003.
- [55] Davidson, S., Crabtree, J., Brunk, B., Schug, J., Tannen, V., Overton, C., Stoeckert, C. "K2/Kleisli and GUS: Experiments in integrated access to genomic data sources," *IBM Systems Journal*, v. 40, n. 2, pp. 512-531, 2001.
- [56] Davidson, S. B., Overton, C., Buneman, P. "Challenges in Integrating Biological Data Sources". *Journal of Computational Biology*. v. 2, n. 4, pp. 557-572, 1995.
- [57] DSSP: Database of Secondary Structure in Proteins. Disponível em: <http://www.sander.ebi.ac.uk/dssp/>, 2002.
- [58] ebXML. Disponível em <http://ebxml.org>, 2002.
- [59] Eclipse Java IDE. Disponível em <http://www.eclipse.org/>.
- [60] Elmasri, R., Navathe, S. *Fundamentals of Databases Systems*, 3rd edition. Addison-Wesley, 2000.
- [61] Entrez Retrieval System for Searching Linked Databases. <http://www.ncbi.nlm.nih.gov/Entrez/>. Acessado em junho de 2002.
- [62] Etzold, T., Ulyanov, A., Argos, P. "SRS: Information Retrieval System for Molecular Biology Data Banks". *Methods Enzymol*, v. 266, pp 114-128, 1996.
- [63] Feitelson, D., Treinin, M. "The Blueprint for Life?" *IEEE Computer*, v. 35, n. 7, pp. 34-40. July 2002.
- [64] Foster, I., Kesselman, C. "A Data Grid Reference Architecture", *Technical Report*, GriPhyN-2001-12, 2001.
- [65] Foster, I., Kesselman, C., Tuecke, S. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations". *International J. Supercomputer Applications*, v. 15, n. 3, pp. 200-222, 2001.
- [66] Foster, I., Voeckler, J., Wilde, M., Zhao, Y. "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation". *International Conference on Scientific and Statistical Database Management*, Edinburgh, Scotland, pp.37-46, 2002.
- [67] Fujibuchi, W., Goto, S., Migimatsu, H., Uchiyama, I., Ogiwara, A., Akiyama, Y., Kanehisa, M. "DBGET/LinkDB: An Integrated Database Retrieval System". *Pacific Symposium. Biocomputing*, pp. 683-694, 1998

- [68] Futuyma, D. J. *Biologia Evolutiva*. Sociedade Brasileira de Genética, 2ª Edição, 1992.
- [69] Gehring, W. J. "Master Control Genes in Development and Evolution: The Homeobox Story (The Terry Lectures)". *Yale University Press*, 1998.
- [70] Genetic Science Learning Center. Disponível em: <http://gslc.genetics.utah.edu>, 2002.
- [71] Georgakopoulos, D., Hornick, M., Sheth, A. "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure". *Distributed and Parallel Databases*, v. 3, n. 2, pp. 119-153, 1995.
- [72] Goodman, N. "Biological Data Becomes Computer Literate: New Advances in Bioinformatics". *Current Opinion in Biotechnology*, v. 13, n. 1, pp. 68-71, February 2002.
- [73] Goodman, N., Rozen S., Stein, L. D., Smith, A. "The LabBase System for Data Management in Large Scale Biology Research Laboratories". *Bioinformatics*, v. 14, n. 7, pp. 562-574, 1998.
- [74] Goodman, N., Rozen S., Stein, L. D. "The LabFlow System for Workflow Management in Large Scale Biology Research Laboratories". *Proceedings of the 6th International Conference on Intelligent Systems for Molecular Biology (ISBM)*, pp. 69-77, Montreal, Quebec, 1998.
- [75] Gorga, Frank R. "Introduction to Protein Structure". Disponível em: <http://webhost.bridgew.edu/fgorga/proteins/default.htm>, 2002.
- [76] GriPhyN Project. Disponível em <http://www.griphyn.org>, 2004.
- [77] Guarino, N. "Understanding, Building and Using Ontologies, *International Journal of Human-Computer Studies*, v. 46, n. 2-3, pp.293-310, 1997.
- [78] Guarino, N., Giaretta, P. "Ontologies and Knowledge Bases: Towards a Terminological Clarification". In *N. Mars (ed.) Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*. IOS Press, Amsterdam, pp. 25-32, 1995.
- [79] Hammer, J., Schneider, M. "Genomics Algebra: A New, Integrating Data Model, Language, and Tool for Processing and Querying Genomic Information." In *Proceedings of the First Biennial Conference on Innovative Data Systems Research*, Morgan Kaufman Publishers, Asilomar, CA, pp. 176-187, January, 2003.
- [80] Head-Gordon, T., Wooley, J. C. "Computational challenges in structural and functional genomics". *IBM Systems Journal*, v. 40, n. 2, pp. 265-296, 2001.
- [81] Heath, L., Ramakrishnan, N. "The Emerging Landscape of Bioinformatics Software Systems". *IEEE Computer*, v. 35, n. 7, pp. 41-45. July 2002.
- [82] Hollingsworth, D. "The Workflow Management Coalition Specification - Workflow Management Coalition. The Workflow Reference Model." Document Number TC00-1003, Document Status - Issue 1.1. 1995.
- [83] Horrocks I. "DAML+OIL: a Reason-able Web Ontology Language". In *Proceedings of EDBT 2002*, March 2002.

- [84] Human Genome Project. Disponível em: <http://www.nhgri.nih.gov/HGP/>, 2002.
- [85] Human Genome Project Information. Disponível em: <http://www.ornl.gov/hgmis/>, 2002.
- [86] Human Genome News. "Human Genome Project Fact Sheet", May 2001.
- [87] IBM BPWS4J. Disponível em <http://www.alphaworks.ibm.com/tech/bpws4j>.
- [88] IBM. "Lotus Workflow". Disponível em <http://www.lotus.com/workflow>
- [89] Interoperable Informatics Infrastructure Consortium – I3C. Disponível em: <http://www.i3c.org/>, 2003.
- [90] J2EE Enterprise JavaBeans Technology. Disponível em: <http://java.sun.com/products/ejb/>
- [91] Java Technology. Disponível em <http://java.sun.com/>, 2002.
- [92] Jornal do Comércio On-line. "Nasce Vitória, Primeiro Clone Animal no Brasil", *Jornal do Comércio*, Recife, 22 de março de 2001. Disponível em http://www2.uol.com.br/JC/_2001/2203/br2203_5.htm, 2004.
- [93] Kim, Junhyong. "Computers are from Mars, Organisms are from Venus: An Interrelationship Guide to Biology and Computer Science". *IEEE Computer*, v. 35, n. 7, pp. 25-32. July 2002
- [94] Kramler, G., and Retschitzegger, W. "Specification of Interorganizational Workflows - A Comparison of Approaches". Institute of Software Technology and Interactive Systems, Business Informatics Group, Vienna University of Technology, Vienna, Austria, Technical Report, 08/02, 2002.
- [95] Kreger, H. "Web Services Conceptual Architecture (WSCA 1.0)". IBM Software Group. Disponível em <http://www-4.ibm.com/software/solutions/webservices/pdf/WSCA.pdf>
- [96] Kuenzl, J. "Development of a Workflow-based Infrastructure for Managing and Executing Web Services". Universität Stuttgart, Fakultät Informatik, Diplomarbeit Nr.1997, 2002.
- [97] Laskowski, R. A., Macarthur, M. W., Moss, D. S., Thornton, J. M. "PROCHECK: a Program to Check the Stereochemical Quality of Protein Structures", *Journal of Appl. Cryst.*, v. 26, pp. 283-291, 1993.
- [98] Laud, A., Bhowmick, S. S., Cruz, P., Singh, D. T., Ramesh, G. "The gRNA: A Highly Programmable Infrastructure for Prototyping, Developing and Deploying Genomics-Centric Applications". In *Proceedings of 28th International Conference on Very Large Data Bases*, pp. 402-409, August 20-23, Hong Kong, China, 2002.
- [99] Letovsky, S.I., Cottingham, R.W., Porter, C.J., Li, P.W.D. "GDB: the Human Genome Database". *Nucleic Acids Research*, v. 26, n. 1, pp. 94-99, 1998.
- [100] Lane, M., Edwards, J. L., Nielsen, E. "Biodiversity Informatics: The Challenge of Rapid Development, Large Databases, and Complex Data". In *Proceedings of 26th International Conference on Very Large Data Bases*, pp. 729-732, September 10-14, 2000, Cairo, Egypt.

- [101] Leite, F., Costa, L. A. G., Baião, F., Mattoso, M. L. Q. "Publishing and Querying Blast Results Using Grid Services". *Revista Tecnologia da Informação*, v. 3, n. 2, pp. 95 - 97, 2003.
- [102] Lemos, M. "Workflow para Bioinformática". Tese de Doutorado, PUC-RIO, Brasil, Agosto de 2003.
- [103] Leymann, F. "Managing Business Processes via Workflow Technology". *In Proceedings of 27th International Conference on Very Large Data Bases*, pp. 729, September 11-14, 2001, Roma, Italy.
- [104] Leymann, F., "Web Services Flow Language (WSFL 1.0)", Disponível em <http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf>, Maio 2001.
- [105] Meyer, L. "Parallel Strategies for Processing Scientific Workflows". COPPE/UFRJ, Brazil, April 2004.
- [106] Meyer, L. Rössle, S., Bisch, P., Mattoso, M. "Parallelism in Bioinformatics Workflows", *in Proc. Int. Conf. VECPAR*, (Valencia, 2004), to appear.
- [107] Microsoft. "The Distributed Component Object Model (DCOM)". Disponível em: <http://www.microsoft.com/com/tech/dcom.asp>
- [108] Microsoft. "MS Message Queuing (MSMQ)". Disponível em: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/messgqueuecomps.asp>
- [109] Mougnot, I., Libourel, T., Déhais, P. "Genetic Sequence Annotation within Biological Databases". *In Proceedings of the Fourth International Conference on Databases Systems for Advanced Applications (DASFAA)*, pp. 333-341, Singapore, April-10-13, 1995
- [110] Muehlen, M. Allen, R. "Workflow Classification Embedded & Autonomous - Workflow Management Systems". Workflow Management Coalition. March 10th, 2000.
- [111] MySQL Database Server. Disponível em: <http://www.mysql.com/products/mysql/>, 2004.
- [112] NCBI - National Center for Biotechnology Information. Disponível em: <http://www.ncbi.nlm.nih.gov/>, 2003.
- [113] Needleman, S., Wunsch, C. "A General Method Applicable to the Search for Similarities in the Amino Acid Sequences of Two Proteins". *Journal of Molecular Biology*, v. 48, pp. 444-453, 1970.
- [114] Object Management Group. "CORBA/IIOP Specification". Disponível em: http://www.omg.org/technology/documents/formal/corba_iiop.htm
- [115] Okayama, T., Tamura, T., Gojobori, T., Tateno, Y., Ikeo, K., Miyasaki, S., Fukami-Kobayashi, K., Sugawara, H.. "Formal Design and Implementation of an Improved DDBJ DNA Database with a New Schema and Object-oriented Library", *Bioinformatics*, v. 14, n. 6, pp. 472-478, 1998.
- [116] Oracle Workflow. Disponível em <http://www.oracle.com/appsnet/technology/products/docs/workflow.html>, 2004.

- [117] Open Grid Services Architecture Data Access and Integration (OGSA-DAI). Disponível em <http://www.ogsadai.org.uk/>, 2004
- [118] OWL-S Web Service Ontology. Disponível em <http://www.daml.org/services/owl-s/>, 2004.
- [119] Özsu, M., Valduriez, P. *Principles of Distributed Database Systems*. 2nd Edition. Prentice Hall, 1999.
- [120] Pallone, S. "Comunidade realiza primeiro encontro da área". Revista Eletrônica de Jornalismo Científico - Bioinformática, no. 46, Agosto 2003. Disponível em <http://www.comciencia.br/reportagens/bioinformatica/bio05.shtml>.
- [121] Patel, C., Supekar, K., Lee, Y. "A QoS Oriented Framework for Adaptive Management of Web Service Based Workflows". In *Proceedings of 14th International Conference on Database and Expert Systems Applications (DEXA)*, pp. 826-835, Prague, Czech Republic, 2003.
- [122] Paton, N., Goble, C. "Information Management for Genome Level Bioinformatics". In *Proceedings of 27th International Conference on Very Large Data Bases*, pp. 728, September 11-14, 2001, Roma, Italy.
- [123] Paton, N. W., Stevens, R., Baker, P. G., Goble, C. A., Bechhofer, S., Brass, A. "Query Processing in the TAMBIS Bioinformatics Source Integration System". In *Proceedings of the 11th International Conference on Scientific and Statistical Databases*, pp. 138-147, IEEE Press, New York (1999).
- [124] Perl Mongers. Disponível em: <http://www.perl.org/>, 2003.
- [125] Persidis, A. "Bioinformatics". *Nature Biotechnology*, v. 17, pp. 828-830, 1999.
- [126] Preuner, G., Schrefl, M. "Integration of Web Services into Workflows through a Multi-Level Schema Architecture". In *Proceedings of 4th IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS'02)*, pp. 51-60, Newport Beach, California, June 26 - 28, 2002.
- [127] RasMol and OpenRasMol. Disponível em <http://www.openrasmol.org/>, 2004.
- [128] Robbins, R. J. "Challenges in the Human Genome Project". *IEEE Engineering in Biology and Medicine*, pp 25-34, March, 1992.
- [129] Rocco, D., Critchlow, T. "Discovery and Classification of Bioinformatics Web Services". *Tech. Report UCRL-JC-149963*, Lawrence Livermore National Lab, USA, 2002.
- [130] Rössle, S. "Desenvolvimento de um Sistema Computacional para Modelagem Comparativa em Genômica Estrutural: Análise de Sequências do Genoma da *Gluconacetobacter Diazotrophicus*". Tese de Doutorado, IBCCF/UFRJ, Brasil, 2004.
- [131] Rössle, S., Carvalho, P., Dardenne, L., Bisch, P. "Development of a Computational Environment for Protein Structure Prediction and Functional Analysis". In *Proceedings of 2nd Brazilian Workshop on Bioinformatics (WOB)*, Rio de Janeiro, 2003.
- [132] Šali, A., 2001, "MODELLER: A Program for Protein Structure Modeling Release 6", Rockefeller University.

- [133] Schnase, John L. "Research Directions in Biodiversity Informatics". In *Proceedings of 26th International Conference on Very Large Data Bases*, pp. 697-700, September 10-14, 2000, Cairo, Egypt.
- [134] Seibel, L. F. "BioAXS: Uma Arquitetura de Dados e Aplicações da Biologia Molecular". Tese de Doutorado, PUC-Rio, Brasil, Abril de 2002
- [135] Singh, M. Vouk, M. "Scientific Workflows: Scientific Computing meets Transactional Workflows". In *Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions*, pp. 28-34, Univ. Georgia, Athens, GA, USA, 1996.
- [136] Smith, T., Waterman, M. "Identification of Common Molecular Subsequences". *Journal of Molecular Biology*, v. 147, pp. 195-197, 1981.
- [137] Spyns, P., Meersman, R., Jarrar, M. "Data Modelling versus Ontology Engineering", *SIGMOD Record*, v. 31, n. 4, pp. 12-17, 2002.
- [138] Steffen, David. "An Introduction to Biocomputing". Version 2.01, October 2001. <http://www.techfak.uni-bielefeld.de/bcd/Curric/Introd/ch0.html>
- [139] Stoesser, G., Baker, W., Broek, A.v.d., Camon, E., Garcia-Pastor, M., Kanz, C., Kulikova, T., Leinonen, R., Lin, Q., Lombard, V., Lopez, R., Redaschi, N., Stoehr, P., Tuli, M.A., Tzouvara, K., Vaughan, R. "The EMBL Nucleotide Sequence Database". *Nucleic Acids Research*, v. 30, pp. 21-26, 2002.
- [140] Sun Microsystems. "Java Remote Method Invocation Specification". Disponível em: <http://java.sun.com/j2se/1.4.2/docs/guide/rmi/spec/rmiTOC.html>
- [141] Suzuki, D. T. et al. *Introdução a Genética*. Guanabara Koogan, 6ª Edição. 1998.
- [142] Teixeira, F. "Fábrica de Web Services para Dados Científicos". *Dissertação de M.Sc.* A ser apresentada, COPPE/Sistemas, UFRJ, Brasil, 2004.
- [143] Teixeira, F., Cavalcanti, M. C., Baião, F., Meyer, L., Rössle, S., Bisch, P., Mattoso, M. "Data Management via Web Services in Bioinformatics Workflows". *Second Brazilian Workshop on Bioinformatics*, Macaé, Rio de Janeiro, Brazil, 2003.
- [144] Thatte, S., "XLANG: Web services for Business Process Design". Disponível em http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm, Microsoft Corporation, 2001.
- [145] The bioJava Project. Disponível em: <http://biojava.org>, 2002.
- [146] The bioPerl Project. Disponível em: <http://bioperl.org/>, 2002.
- [147] The bioPython Project. Disponível em: <http://biopython.org/>, 2002.
- [148] The Collaborative Electronic Notebook Systems Association - CENSA. Disponível em <http://www.censa.org/>, 2004.
- [149] The European Molecular Biology Open Software Suite – EMBOSS. Disponível em: <http://www.emboss.org/>, 2003.
- [150] The Institute for Genomic Research. Disponível em: <http://www.tigr.org/>, 2002.

- [151] The Internet Society, Network Working Group, RFC: 2616. "Hypertext Transfer Protocol - HTTP/1.1". Disponível em <ftp://ftp.isi.edu/in-notes/rfc2616.txt>, 1999.
- [152] The Open Bioinformatics Foundation. Disponível em: <http://open-bio.org/>, 2003.
- [153] The Workflow Management Coalition. Disponível em: <http://www.wfmc.org>, 2003.
- [154] UDDI Spec Technical Committee Specification. "The UDDI Version 3.0.1 Specification". Disponível em <http://www.uddi.org/specification.html>, 2002.
- [155] U.S. Department of Energy, Human Genome Program. "2001 Primer on Molecular Genetics". October 2001.
- [156] U.S. Department of Energy, Human Genome Program, "Genomics and Its Impact on Science and Society: A 2003 Primer". 2003.
- [157] U.S. DOE Genome Image Gallery. Disponível em <http://www.doegenomes.org/>, 2004.
- [158] Virdell, M. "Business processes and workflow in the Web services world". IBM developerWorks, 1 January 2003. Disponível em <http://www-106.ibm.com/developerworks/webservices/library/ws-work.html>.
- [159] VisualScript XML. Disponível em <http://www.visualscript.com/>.
- [160] W3C. "Web Services Architecture", Working Draft, Nov. 2002
- [161] W3C Note. "Web Services Description Language (WSDL) 1.1". Disponível em: <http://www.w3.org/TR/wsdl.html>, 2001.
- [162] W3C Note on Simple Object Access Protocol (SOAP) 1.1, Disponível em: <http://www.w3.org/TR/SOAP/>, 2000.
- [163] W3C (World Wide Web Consortium) Note, "Web services Conversation Language (WSCL) 1.0". Disponível em <http://www.w3.org/TR/2002/NOTE-wscl10-20020314/>, Março 2001.
- [164] W3C (World Wide Web Consortium) Recommendation, "Extensible Markup Language (XML) 1.0 (Second Edition)". Disponível em <http://www.w3.org/TR/REC-xml>, 2000.
- [165] W3C (World Wide Web Consortium) Recommendation. "OWL Web Ontology Language Reference". Disponível em <http://www.w3.org/TR/owl-ref/>, 2004.
- [166] W3C (World Wide Web Consortium) XML Specification ("XMLspec") DTD, Version 2.1. Disponível em <http://www.w3.org/XML/1998/06/xmlspec-report.htm>, 2001.
- [167] W3C (World Wide Web Consortium). Disponível em: <http://www.w3c.org>
- [168] Washington University Basic Local Alignment Search Tool Version 2.0, WU-BLAST. Disponível em <http://www.ebi.ac.uk/blast2/>, 2004.
- [169] Watson, J. D., Crick, F. H. C. "Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid". *Nature*, v. 171, pp. 737-738, 1953.

- [170] WebserviceX.NET. "Global Weather". Disponível em <http://www.webservicex.net/WS/WSDetails.aspx?CATID=12&WSID=56>, 2004.
- [171] Weske, M., Vossen, G., Medeiros, C. "Scientific Workflow Management: WASA Architecture and Applications". In *Fachbericht Angewandte Mathematik und Informatik*, 03/96-I, 1996.
- [172] Westbrook, J., Feng Z., Chen L., Yang, H., Berman, H. "The Protein Data Bank and Structural Genomics". *Nucleic Acids Research*, v. 31, n. 1, pp. 489–491, 2003.
- [173] Wiederhold, G., "Mediation in Information Systems", *ACM Computing Survey*, v. 27, n. 2, pp. 265-267, 1995.
- [174] Wilbur, W., Lipman, D. J. "Rapid Similarity Searches of Nucleic Acid and Protein Data Banks". In *Proceedings of the National Academy of Sciences*, v. 80, pp. 726–730, 1983.
- [175] Wilkinson, M. D., Links, M. "BioMOBY: an Open-source Biological Web Services Proposal". *Briefings In Bioinformatics*, v. 3, n. 4, pp. 331-341, 2002.
- [176] Wilkinson, M. D., Gessler, D., Farmer, A., Stein, L. "The BioMOBY Project Explores Open-Source, Simple, Extensible Protocols for Enabling Biological Database Interoperability". In *Proc Virt Conf Genom and Bioinf*, v. 3, pp. 16-26, 2003.
- [177] Wilmut, I., Schnieke, A. E., McWhir, J., Kind, A. J., Campbell, K. H. S. "Viable Offspring Derived from Fetal and Adult Mammalian Cells". *Nature*, v. 385, pp. 810-813, February 27, 1997.
- [178] Workflow Patterns. Disponível em <http://tmitwww.tm.tue.nl/research/patterns/>, 2004.
- [179] Wroe, C., R. Stevens, Goble, C., Roberts, A. and Greenwood, M. "A suite of DAML+OIL ontologies to describe bioinformatics web services and data". *International Journal of Cooperative Information Systems*, v. 12, n. 2, pp. 197-224, 2003.
- [180] Wu, C.H., Huang, H., Arminski, L., Castro-Alvear, J., Chen, Y., Hu, Z., Ledley, R.S., Lewis, K.C., Mewes, H., Orcutt, B.C., Suzek, B.E., Tsugita, A., Vinayaka, C. R., Yeh, L.L., Zhang, J., Barker, W.C. "The Protein Information Resource: an integrated public resource of functional annotation of proteins". *Nucleic Acids Research*, v. 30, pp. 35-37, 2002.
- [181] Zaha, A. et al. *Biologia Molecular Básica*. Mercado Aberto, 3a Edição. Porto Alegre, 2003.
- [182] Xu Y., Xu, D. "Protein Structure Prediction by Protein Threading and Partial Experimental Data". In *Current Topics in Computational Biology*, edited by Jiang, T., Xu, Y., Zhang, M. Q. The MIT Press. Cambridge, Massachusetts, pp. 467-502, 2002.