"GERÊNCIA DE MAPEAMENTO ENTRE ESQUEMAS DE REPRESENTAÇÃO DE DADOS GENÉTICOS"

Fátima Cristina Vieira Gonçalves

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:

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

Prof. Nelson Francisco Favilla Ebecken, D.Sc.

Profª. Ana Maria de Carvalho Moura, Dr.

RIO DE JANEIRO, RJ – BRASIL

JUNHO DE 2004

GONÇALVES, FÁTIMA CRISTINA VIEIRA

Gerência de Mapeamento entre Esquemas de Representação de Dados Genéticos [Rio de Janeiro] 2004

VIII, 94 p., 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2004)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Banco de Dados
2. Mapeamento de Esquemas
3. Representação de Dados Genéticos

I. COPPE/UFRJ II. Título (série)

Agradecimentos

Aos meus pais que me deram o apoio necessário para chegar até aqui.

À minha irmã que sempre me fazia rir mesmo nos momentos mais complicados e renovava minhas energias para seguir em frente.

Ao Fábio que sempre foi a minha força. Sem ele, nada disso teria sido possível.

À minha orientadora Marta Mattoso por ter sido uma ótima orientadora, sempre sugerindo novas idéias e contribuindo muito com sua experiência. Agradeço também por não deixar de acreditar que eu conseguiria concluir minha tese, mesmo com todas as dificuldades para conciliar a vida acadêmica com a profissional.

Ao Blaschek que foi o grande incentivador para que eu ingressasse no mestrado e sempre tinha uma palavra amiga.

À Maria Cláudia Cavalcanti que me ajudou bastante a organizar minhas idéias durante vários momentos da pesquisa da minha tese.

À Fernanda Baião que contribuiu muito com novas idéias principalmente para a apresentação do meu trabalho.

Ao Sérgio Lifschitz e Luiz Fernando Seibel que me incentivaram a trabalhar na área de gerência de dados em bioinformática.

À Patrícia que sempre ajudava a resolver os problemas com a maior presteza.

Ao CNPq pelo apoio financeiro.

Acima de tudo, agradeço a Deus por ter conseguido enfrentar todos os obstáculos e concluir esse trabalho com sucesso.

A todas as outras pessoas que de uma forma ou de outra contribuíram para a execução desse trabalho.

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

GERÊNCIA DE MAPEAMENTO ENTRE ESQUEMAS DE REPRESENTAÇÃO DE DADOS GENÉTICOS

Fátima Cristina Vieira Gonçalves

Junho / 2004

Orientadora: Marta Lima de Queirós Mattoso

Programa: Engenharia de Sistemas e Computação

Existem muitos desafios na área de dados em bioinformática, principalmente com relação à grande quantidade de dados que são produzidos em diferentes formatos e modelos. Há basicamente duas abordagens para interagir com estas fontes de dados genéticos heterogêneas. A primeira abordagem é baseada em um mapeamento para um esquema padrão. Esta representação global pode ser usada para mapear todos os dados disponíveis para um esquema único. A segunda abordagem trabalha com vários mapeamentos específicos. Esta última abordagem não força o mapeamento para um esquema padrão permitindo a convivência de diversos esquemas. A integração é feita de acordo com os requisitos específicos da atividade que está sendo tratada no momento.

Nosso trabalho se encaixa dentro dessa segunda abordagem junto a ambientes integrados para trabalhar com as diversas fontes de dados genéticos em seus formatos originais e seus *workflows*. Nesses ambientes é necessário que o biólogo informe as regras de mapeamento que possibilitam a integração desejada. Na maioria das vezes, o mapeamento dessas informações não é trivial e realmente depende da interação com o biólogo. Nossa proposta é oferecer apoio ao biólogo durante o processo de mapeamento das informações genéticas, ajudando na integração de diferentes fontes de dados e/ou entradas e saídas de programas. Foi desenvolvida uma ferramenta que pode ser utilizada em conjunto com arquiteturas que disponibilizam aos biólogos um ambiente integrado de armazenamento e análise dos dados genéticos independentemente dos formatos em que estão originalmente disponíveis.

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

MAPPING MANAGEMENT BETWEEN GENETIC SCHEMA

Fátima Cristina Vieira Gonçalves

June / 2004

Advisor: Marta Lima de Queirós Mattoso

Department: Computer and Systems Engineering

There are many challenges within bioinformatics data, mainly with respect to the vast amount of data being produced in different formats and models. There are basically two approaches to interoperate with these heterogeneous sources of genetic data. The first approach is based on a global schema mapping. This global representation can be used to map all the different available data into a unique schema. The second approach works with several specific schema mappings. This latter approach does not force mappings to one global schema that encompasses all possible schemas. Rather, the integration is made depending on the current activity requirements.

Our work fits this second approach within integrated environments that work with several sources of genetic data in its original format and workflows. With these environments it is necessary that the biologist inform the mapping rules that allow the desired integration. Most of the time, the mapping of these informations is not trivial and depends on the interaction of the biologist. Our proposal is to provide support to the biologist during the mapping process of the genetic information, helping on the integration of different sources of data and the programs inputs and outputs. We develop a tool that can be plugged to architectures that give biologists an integrated environment to store and analyze the genetic data independent of their formats, in which they are originally available.

Índice

Índice de Figuras e Tabelas

CAPÍTULO 1. INTRODUÇÃO

Em alguns setores de pesquisa em bioquímica e biologia molecular, o uso da informática é vital para que se possa alcançar mais resultados a partir da análise dos dados gerados na pesquisa biomédica com dados experimentais. Com a explosão de informações obtidas disponíveis aos pesquisadores, novas ferramentas são necessárias para analisar os dados de seqüências e do mapa do genoma. Reconhecer o início, o fim de cada gene e obter o completo alinhamento das seqüências requer extensivas comparações para a identificação de similaridades. Além disto, a comparação de seqüência entre espécies também pode ajudar a elucidar similaridades, revelando aspectos no funcionamento e estrutura desses genes. Agora, o desafio principal, é incrementar os programas de entrada, acesso, análise e manipulação dos diversos tipos de dados biológicos para os sistemas utilizados nos diferentes laboratórios.

Existem muitos problemas a serem estudados, principalmente devido ao fato de que uma vasta quantidade de dados genéticos está sendo produzida [1, 2, 7] em diferentes formatos e modelos [3, 4, 5, 13], mas que estão conectados por algum processo. Podemos citar como exemplo os experimentos científicos *in silico* [62]. Os recursos usados nesses experimentos são principalmente programas e dados. É importante notar que estas aplicações científicas envolvem programas encadeados, o que significa que vários programas são executados de uma maneira organizada. Este encadeamento de programas é chamado de *workflow* científico. Uma vez definidos, estes workflows podem ser reusados por outras aplicações, o que os transforma em recursos científicos valiosos [45]. Em ambientes de bioinformática, a composição do *workflow* é uma operação freqüente, e muito complexa para ser controlada. Para construir um experimento, o cientista enfrenta várias dificuldades, tais como encontrar qual o programa ideal para sua necessidade, ajustar os parâmetros adequados, gerenciar os dados de entrada e saída, construir e reutilizar os *workflows*. Tipicamente, estes *workflows* são executados usando *scripts* devido a sua facilidade de utilização, apesar das suas particularidades e dificuldade de reutilização. Um exemplo de uma aplicação com essas características é o MHOLline [66, 62] que combina um conjunto específico de programas. Para a identificação de estruturas, o MHOLline usa o algoritmo de busca BLAST [14]. Um passo de refinamento das seqüências selecionadas pelo BLAST é implementado pelo programa BATS. Um terceiro programa, MODELLER, é responsável pelo alinhamento automatizado e construção do modelo e os modelos 3D resultantes são avaliados usando o programa PROCHECK. Os dados intermediários resultantes da execução desse *workflow* precisam ser armazenados e muitas vezes mapeados para outros formatos para que possam ser reutilizados em outro *workflow*, ou em outros programas do próprio *workflow*.

Cabe então, pesquisar maneiras de armazenar de forma organizada e otimizada os dados genéticos e propor soluções para os diversos tipos de integrações ou mapeamentos necessários entre as informações e processos dessa área. Essa integração não é trivial, pois as várias fontes de dados genéticos podem possuir qualquer tipo de representação e o mapeamento nem sempre pode ser automatizado. A primeira dificuldade nessa integração é identificar o mapeamento necessário. Nesse caso a interação com o usuário é necessária para definir o mapeamento adequado entre as informações. A segunda dificuldade é, uma vez definido qual o mapeamento a ser feito, realizar a transformação propriamente dita. Na maioria das vezes, isso implica em escrever um programa específico para essa transformação. Essas dificuldades se repetem sempre que um novo *workflow* é definido.

O *workshop* XEWA-00 [12] mostrou claramente o interesse com os problemas relacionados à informática na pesquisa genética. Durante este *workshop*, os membros da comunidade de bioinformática concluíram que XML (*Extensible Mark-Up Language* [22, 60]) poderia simplificar o acesso às grandes e heterogêneas fontes de dados genéticos que estão distribuídas pela Web, e que deveria ser o padrão adotado por causa de suas facilidades para o processo da troca de informação. De fato, alguns softwares de bioinformática já geram dados no formato XML, como por exemplo, o NCBI BLAST [14].

Há basicamente duas abordagens para tratar esta variedade de fontes de dados genéticos. Essas abordagens são oriundas de trabalhos de integração de SGBDs e bases de dados heterogêneas [67, 68, 69]. A primeira abordagem é baseada em um mapeamento das várias representações de dados genéticos para um esquema global único. Esta representação global é a união de todos os diferentes esquemas disponíveis em um único esquema, ou seja, são definidas representações de elementos do mundo real e todas as informações desses elementos devem seguir essa representação. Nesse caso todos as fontes de dados seriam convertidas para esse modelo global e os programas seriam adaptados para funcionar com esse mesmo modelo. Com a adoção de um modelo único global, o acesso aos dados fica mais rápido e pode ser otimizado de modo a agilizar as análises, o processo de armazenamento e a recuperação. Além disso, as informações e os programas são facilmente reaproveitados para novas análises. O grande problema com esse tipo de abordagem é conseguir um consenso com a comunidade científica de qual modelo seria indiscutivelmente mais adequado para ser adotado como um modelo único global. Além disso, deve-se levar em conta o esforço necessário para se converter as fontes de dados atuais para esse novo modelo global.

A segunda abordagem trabalha com diversas representações em seus formatos originais e necessita de mapeamentos para esquemas específicos. Esta abordagem não força o

mapeamento para um esquema único. Ao contrário disso, integra dependendo dos requisitos da atividade corrente. Ou seja, cada grupo continua trabalhando com seus esquemas específicos e, de acordo com a necessidade, precisa realizar mapeamentos para possibilitar a integração com as informações de outros grupos que trabalham com esquemas diferentes. A grande vantagem dessa abordagem é não obrigar a utilização de um esquema único causando impactos nas rotinas utilizadas atualmente pelos diversos grupos de pesquisa. A desvantagem é que necessita de um ambiente que dê suporte no processo de integração, onde o usuário precisa definir as informações de mapeamento e ter uma ferramenta que realize as transformações necessárias.

Alguns trabalhos (PROXIML [33], GAME [39] e AGAVE [40]) seguem a primeira abordagem e aproveitam a vantagem do XML ser uma linguagem amplamente utilizada para troca de informações, para integrar dados genéticos heterogêneos. Esses trabalhos têm como objetivo representar os dados genéticos através de um esquema global único no formato XML e usa as facilidades de troca de informações inerentes ao XML para minimizar as dificuldades na adoção dessas propostas.

Também adotando a abordagem de um modelo conceitual global, temos o modelo proposto por Paton em [41] e o projeto GIMS [64, 65]. O projeto DataFoundry desenvolvido no LLNL (Lawrence Livermore National Laboratory) [44], usa metamodelos e mediadores para representar globalmente as bases de dados. O projeto myGrid [42, 43] apresenta um modelo global mais flexível, baseado em ontologias e em ferramentas para lidar com os diferentes domínios das ontologias.

É difícil encontrar um consenso sobre o uso de um modelo único para todas as bases de dados genéticas, porque cada grupo de pesquisa possui sua própria representação e é difícil encontrar um modelo que satisfaça todas as necessidades de cada grupo. Com isso, os trabalhos que seguem a abordagem de integrações específicas estão começando a aparecer. A utilização do XML nesses casos oferece vantagens por ser um padrão para troca de dados, mas não fornece o apoio semântico para a integração.

Um trabalho pioneiro nesta segunda abordagem, que adota as diversas representações em seus formatos originais, é o "*Genome Database Framework*" [6] desenvolvido na PUC-Rio, também chamada de integração virtual. Este *framework* suporta os grupos de pesquisa que geram e processam dados heterogêneos. Foca na integração da informação genética que é espalhada em um ambiente distribuído (principalmente na Web). A integração fornecida em [6] é feita sob encomenda e oferece muitas opções que podem ajudar em questões de modelagem e de integração. Também na linha da integração virtual, foi desenvolvida a

arquitetura SRMW (*Scientific Resources Management based on Web Services*) [37, 45], mas com foco em composições de programas. Ambos os trabalhos [6, 37] oferecem flexibilidade uma vez que não forçam os usuários a adotar um modelo único de representação de dados genéticos. Enquanto [6, 37] cuidam de aspectos semânticos e genéricos de integração, surge a necessidade de um apoio específico à integração que facilite o mapeamento e a conversão de esquemas. O usuário é quem detém o conhecimento de como deve ser feito o mapeamento das informações, mas esbarra no problema de como realizar efetivamente a conversão. Complicando mais esse cenário, temos o fato de que esses mapeamentos são bastante dinâmicos e, para atender esse ritmo, são necessárias alterações constantes nas ferramentas de conversão. Para dar suporte a essa necessidade do usuário, muitas vezes é necessário desenvolver novas ferramentas para cada caso. O problema que nos propomos a resolver é dar suporte ao usuário durante o processo de mapeamento e gerar automaticamente mecanismos para a transformação propriamente dita.

Visando essa motivação, nosso trabalho se encaixa nas duas abordagens descritas. Na primeira abordagem, para se trabalhar com um esquema único global, será preciso mapear as informações nos formatos originais para o esquema desejado. Na segunda abordagem, a idéia é se trabalhar com as informações em seus formatos originais, mas precisamos realizar mapeamentos entre essas informações para tornar possível a integração. Nosso trabalho está mais focado na segunda abordagem, pois nesse caso o problema de mapeamento é mais complexo devido à diversidade de formatos e à dinâmica dos *workflows*.

Dentro desse contexto, focamos em arquiteturas do tipo [6, 37], que trabalham com a integração de dados e composição de programas científicos, contribuindo para fornecer mapeamentos entre os diversos esquemas interagindo com o usuário de uma maneira flexível. Nossa ferramenta suporta a conversão entre representações de esquemas de bases de dados biológicas, uma vez identificadas as correspondências, o que facilita a integração que o usuário necessita. Esta ferramenta pode ser acoplada a outras para dar ao usuário uma visão integrada para armazenamento e análise dos dados genéticos independentemente dos formatos em que estão originalmente disponíveis. Isso permite ao usuário interagir com os diferentes esquemas de dados genéticos a fim de facilitar o mapeamento visual entre eles.

O mapeamento e a conversão de dados e programas científicos podem ser feitos fixando as regras que serão adotadas previamente ou definindo dinamicamente as ações que devem ser aplicadas. Existem ferramentas disponíveis que realizam esse mapeamento de modo fixo, ou seja, são ferramentas de conversão que transformam uma determinada representação em uma outra. Alguns exemplos desse tipo de ferramenta podem ser

encontrados na Web em sítios de bancos genéticos [3, 4, 5] ou de propostas de representações únicas [33, 34, 35, 36]. A vantagem desses tipos de ferramentas é que estão prontas para serem utilizadas e a intervenção do usuário não é necessária. O problema é a necessidade de construção de uma nova ferramenta ou adaptação de uma existente cada vez que os tipos de representações mudarem. Além disso, obriga o usuário a utilizar o mapeamento já definido na ferramenta sem nenhuma adaptação.

Devido às limitações apresentadas, optamos por propor uma solução dinâmica que desse ao usuário um ambiente mais flexível, que permitisse a realização de vários mapeamentos independentemente das representações utilizadas como entradas. Como resultado, essa dissertação apresenta uma ferramenta que, além de suportar o mapeamento e a conversão entre os diversos esquemas, apoia o usuário na criação de um novo esquema a partir das entradas fornecidas, agregando informações que podem ser relevantes ao problema tratado pelo usuário. Ao mesmo tempo, nossa ferramenta pode ser acoplada a arquiteturas de integração semântica de dados.

Nossa ferramenta possui uma interface visual e ajuda o usuário a realizar o mapeamento das informações genéticas. A partir do mapeamento apontado pelo usuário, a ferramenta gera um arquivo de comandos para realizar automaticamente a transformação propriamente dita.

Para avaliarmos nosso trabalho, criamos um cenário real possível de utilização, acoplando nossa ferramenta com a arquitetura SRMW [37, 45]. O usuário usa a arquitetura SRMW para gerenciar os metamodelos de seus dados, programas e *workflows*. Dependendo do *workflow* definido, será preciso realizar transformações nos dados ou nas entradas e/ou saídas dos programas no momento da instanciação desse *workflow*. Nossa ferramenta ajuda o usuário na definição dos mapeamentos e realização das conversões. Nosso trabalho foi apresentado no Second Brazilian Workshop on Bioinformatics, da SBC, WOB'03 [61], descrevendo suas principais características.

Esta dissertação é organizada como segue. O capítulo 2 descreve o contexto e motivação do trabalho dando uma visão geral das bases de dados utilizadas na bioinformática e descrevendo os principais desafios relacionados à área de gerência de dados em bioinformática. No capítulo 3 descrevemos os principais projetos de gerência de fontes heterogêneas de dados e programas genéticos na área da bioinformática. No capítulo 4 são apresentados a proposta da nossa ferramenta de suporte ao mapeamento de esquemas genéticos e os detalhes da implementação e utilização da ferramenta desenvolvida. Também será apresentado neste capítulo um estudo de caso evidenciando uma das aplicações práticas

da ferramenta, servindo de validação para o nosso trabalho. Finalmente, o capítulo 5 conclui esse trabalho. Os relatórios técnicos [22, 60] e apêndice A explicam alguns conceitos técnicos básicos envolvidos na apresentação desse trabalho. O apêndice B mostra os principais diagramas da ferramenta desenvolvida.

CAPÍTULO 2. BASES DE DADOS NA BIOINFORMÁTICA

Nesse capítulo analisamos o contexto de motivação do trabalho dando uma visão geral das bases de dados na bioinformática e descrevendo os principais desafios relacionados à área de gerência de dados em bioinformática. Na primeira seção é apresentada uma explicação geral dos objetivos da área de bioinformática e dos grupos de pesquisa que atuam nesse meio. Na última seção apresentamos como as informações genéticas são tratadas e quais são os principais problemas e desafios relacionados ao tratamento dessas informações.

2.1. Visão Geral da Área de Dados em Bioinformática

No mundo inteiro, biólogos estão colaborando em um esforço comum para determinar os genomas completos de diversas espécies, entre elas, a da espécie humana, no que é conhecido como o Projeto do Genoma Humano [1, 2]. O mapeamento do genoma humano pretende revolucionar práticas médicas e estudos biológicos e através do estudo das estruturas químicas e das funções dos genes, espera-se entender os dados genéticos replicados pela evolução das gerações e associar genes a diversas doenças como diabetes e alguns tipos de câncer a fim de produzir medidas preventivas e tratamentos mais efetivos.

Mesmo com todos os benefícios que se /esperam, os projetos de biologia molecular se tornaram desafios por se tratar de lidar com grandes quantidades de seqüências de DNA e proteínas, além de outros tipos de informação. Além disso, com a modernização das técnicas utilizadas por esses biólogos, a quantidade desses dados cresceu ainda mais, mostrando-se uma ótima oportunidade para sistemas gerenciadores de bancos de dados (SGBDs).

Estima-se que o conjunto completo de genes necessários para recriar os humanos é de aproximadamente 100.000 genes, distribuídos ao longo de 23 pares de cromossomos, em três a quatro bilhões de nucleotídeos.

O total desses dados biológicos produzidos deve ser primeiramente coletado, armazenado e distribuído em algum tipo de base de dados. Toda esta vasta quantidade de dados requer o uso intenso de sistemas computacionais. Isso faz com que, um dos principais focos do projeto do genoma humano seja a manipulação eficiente e organizada de uma grande massa de dados, motivando o desenvolvimento de bancos de dados para esse propósito.

Novas ferramentas também são necessárias para analisar os dados de seqüências e do mapa do genoma. Reconhecer o início e fim de cada gene e obter o completo alinhamento das seqüências requer extensivas comparações para a identificação de similaridades

(homologias). Além disto, a comparação de seqüência entre espécies, como os ratos, também pode ajudar a elucidar similaridades, revelando aspectos no funcionamento e estrutura desses genes.

Agora, o desafio principal é gerenciar e incrementar os programas de entrada, acesso, análise e manipulação dos diversos tipos de dados biológicos para os vários sistemas de computadores utilizados nos diferentes laboratórios.

Cada vez mais, tem sido intensa a necessidade da utilização da informática para a análise de dados gerados na pesquisa biomédica. Em alguns setores de pesquisa em bioquímica e biologia molecular, o uso da informática é importante na geração de resultados a partir dos dados experimentais. A análise de genomas e proteínas, bem como os estudos relativos à relação estrutura e função de proteínas e de outras macromoléculas de interesse biológico são os setores em que isso mais se evidencia atualmente. A aplicação de técnicas computacionais nas ciências biológicas foi definida sob o termo bioinformática, e hoje é utilizado para identificar diversos assuntos, desde inteligência artificial a algoritmos de análise de genoma.

A existência da Internet e de microcomputadores cada vez mais possantes e ligados em clusters colaboraram muito para o crescimento da bioinformática. De fato, a disseminação de microcomputadores conectados à grande rede (*grids*) permitiu o uso por um número cada vez maior de laboratórios e pesquisadores, de recursos antes disponíveis apenas a certos centros de pesquisas equipados com recursos de supercomputação.

Um dos objetivos principais da bioinformática é racionalizar as informações de seqüências biológicas e projetar ferramentas de análise mais poderosas. O maior guia deste processo deve ser a necessidade de converter seqüências de informação em conhecimento bioquímico e biofísico para decifrar a estrutura, função e indícios evolutivos codificados nessas seqüências.

2.2. Desafios Relacionados à Área de Dados em Bioinformática

Nesta seção serão apresentados os principais problemas e desafios relacionados à área de bioinformática. Vamos apresentar as características bem particulares dos dados biológicos e como os dados biológicos estão normalmente estruturados para serem utilizados pela área de bioinformática.

2.2.1. Características dos Dados Biológicos

Os dados biológicos exibem muitas características especiais que tornam o gerenciamento de informações biológicas um desafio em particular. A seguir estão listadas as principais características conforme levantamento feito por Amabis [50].

- Dados biológicos são mais complexos quando comparados com a maioria dos outros domínios de aplicações. As definições destes dados devem representar subestruturas complexas assim como relacionamentos entre elas. Também existe a necessidade de armazenar o contexto desses dados para a correta interpretação biológica.
- A quantidade e a variedade dos dados é grande implicando que sistemas biológicos devem ser flexíveis no tratamento de tipos de dados e valores.
- Esquemas das fontes de dados biológicos variam rapidamente. Essa característica leva a relançamentos constantes de novos bancos com várias modificações nos esquemas e necessita que os bancos tenham a habilidade de estender esquemas facilmente além de incluir alguma informação sobre diferentes versões dos dados.
- Representação de um mesmo dado por diferentes biólogos podem ser diferentes (mesmo utilizando o mesmo sistema). Logo, mecanismos para alinhar diferentes versões dos dados devem ser suportados.
- A maioria dos usuários não necessita de acesso para escrita ao banco, apenas acesso de leitura.
- Definir e representar consultas complexas são extremamente importantes para os biólogos e por isso, os sistemas propostos para dar suporte aos dados genéticos devem suportá-las sem que seja necessário nenhum conhecimento por parte deles da estrutura e do esquema interno dos dados.
- Usuários das informações biológicas geralmente requerem acesso a valores antigos dos dados, particularmente resultados de verificações anteriores. Por isso, mudanças nos valores devem ser suportadas e ambas as versões recentes e antigas devem ser mantidas.

Devido a essas características apresentadas, os dados biológicos necessitam de um tratamento diferenciado de modo a otimizar o armazenamento e a aplicação de algoritmos e *workflows* utilizados na área de bioinformática. Na próxima seção veremos como os grupos de pesquisa lidam com esses dados, utilizando o conceito de bancos de dados ou simplesmente adotando tipos de arquivos com formatos pré-definidos.

2.2.2. Bancos de Dados e Tipos de Arquivos Genéticos

As pesquisas genéticas resultam em muitas informações que precisam ser armazenadas e processadas. Novas pesquisas utilizam os resultados de pesquisas anteriores para realizar comparação de dados e chegar a novos resultados. Com isso, surgiu o problema de como lidar com grande volume de dados das seqüências de DNA e de proteínas, além de outros dados relacionados como as anotações [13]. Como essas informações estão no formato de texto, a forma inicial de armazenamento foi arquivos texto. O crescimento do tamanho desses dados e as tecnologias disponíveis motivaram o uso de SGBDs.

A tecnologia de bancos de dados, mesmo já presente nesse contexto, ainda não utiliza todas as funcionalidades de um SGBD. Mesmo assim, a maioria dos usuários ainda trabalha com arquivos texto disponíveis nos repositórios públicos e o acesso aos dados é feito através de programas *ad-hoc* como o BLAST [14]. Existem muitas bases de dados de biologia molecular, também conhecidos como bancos de dados genoma, como o *GenBank Sequence Database* [3] (sistema baseado em arquivos texto com uma organização semi-estruturada), o *Annotated Protein Sequence Database (Swiss-Prot)* [5] (sistema relacional) e o *A C. Elegans Database (AceBase)* [4] (sistema orientado a objetos).

É importante notar que muitos dos chamados bancos de dados não são sistemas completos de banco de dados, mais sim sistemas de arquivos com sua própria estrutura de armazenamento, manipulação e métodos de acesso. Esses bancos de dados (ou fontes de dados, em geral) representam atualmente somente uma fração do domínio de conhecimento.

Cada um desses bancos genéticos possui seus próprios esquemas e algoritmos de análise dos dados. Cada pesquisador precisa adaptar suas informações genéticas de acordo com o esquema do banco que deseja publicar suas descobertas. O grande problema é que cada banco genético se torna uma ilha de informações e essas ilhas não se comunicam entre si. No decorrer dessa seção vamos apresentar os principais bancos de dados e tipos de arquivos genéticos utilizados atualmente e seus formatos, seus objetivos e como lidam com as características inerentes aos dados genéticos.

2.2.2.1. Genbank

Atualmente, o Genbank [3] é o principal banco de dados de seqüências genéticas do mundo e é mantido pelo *National Center for Biotechnology Information* (NCBI) [14]. Ele foi estabelecido em 1978 como um repositório central para os dados biológicos. O tamanho do banco de dados tem dobrado aproximadamente a cada 18 meses nos últimos cinco anos.

O sistema é mantido como uma combinação de arquivos *flat*, banco de dados relacionais e arquivos contendo *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 das seqüências pelos autores. Agora, os dados também são recebidos de outros grandes centros de sequenciamento que possuem uma colaboração internacional com o GenBank, como o *EMBL Data Library* do Reino Unido e o *DNA Database of Japan* (DDBJ). Os dados são trocados diariamente para garantir que ambos os três *sites* possuam um completo conjunto de dados de seqüências de DNA.

- Organização do Genbank

Mais de 30.000 diferentes espécies são representadas no GenBank e novas espécies são adicionadas a taxa de 600 por mês. Entradas para humanos constituem 57% do total dessas seqüências. Cada entrada no GenBank inclui uma descrição concisa da seqüência, do nome científico e da taxinomia do organismo e uma tabela de características (*features*) que identificam regiões codificadas e outros aspectos de significado biológico como unidades de transcrição, limites de *intron/exon* e aspectos de mutações e modificações. Traduções de proteínas para regiões de codificação também estão na tabela de características. A primeira linha, além do nome *Locus*, apresenta informações da quantidade de pares de bases de nucleotídeos (484 bp), o tipo da molécula (DNA), a divisão no GenBank (BCT – *bacterial sequences*) e a data da última modificação da seqüência. Referências bibliográficas são incluídas com a ligação para os resumos e artigos da biblioteca da MEDLINE para todas as seqüências que foram publicadas. A seguir é apresentado um trecho de uma entrada do GenBank para a bactéria ANTHRAX [3].

```
LOCUS       BAN413939     484 bp    DNA             BCT       09-OCT-2001
DEFINITION  Bacillus anthracis pagR gene, isolate IT-Carb3-5419.
ACCESSION   AJ413939
VERSION     AJ413939.1  GI:16031499
KEYWORDS    pagR gene.
SOURCE      Bacillus anthracis.
  ORGANISM  Bacillus anthracis
            Bacteria; Firmicutes; Bacillus/Clostridium group;
            Bacillus/Staphylococcus group; Bacillus; Bacillus cereus group.
REFERENCE   1  (bases 1 to 484)
  AUTHORS   Adone,R., Pasquali,P., La Rosa,G., Marianelli,C., Muscillo,M.,
            Fasanella,A., Francia,M. and Ciuchini,F.
  TITLE     Sequence analysis of the genes encoding for the major virulence
            factors of bacillus Anthracis vaccine strain 'Carbosap
  JOURNAL   Unpublished
REFERENCE   2  (bases 1 to 484)
  AUTHORS   Muscillo,M.
  TITLE     Direct Submission
  JOURNAL   Submitted (11-SEP-2001) Muscillo M., Environmental Hygiene,
            Istituto Superiore di Sanita, Viale Regina Elena 299, Rome 00199,
            Italy
FEATURES             Location/Qualifiers
     source          1..484
                     /organism="Bacillus anthracis"
                     /plasmid="pXO1"
                     /strain="Carbosap"
                     /isolate="IT-Carb3-5419"
                     /db_xref="taxon:1392"
                     /country="Italy"
                     /note="vaccine against anthrax in cattle and sheep"
     gene            50..349
                     /gene="pagR"
     CDS             50..349
                     /gene="pagR"
                     /function="regulator of virulence gene transcription"
                     /codon_start=1
                     /transl_table=11
                     /product="PagR protein"
                     /protein_id="CAC93937.1"
                     /db_xref="GI:16031499"
                     /translation="MTVFVDHKIEYMSLEDDAELLKTMAHPMRLKIVNELYKHKALNV
                     TQIIQILKLPQSTVSQHLCKMRGKVLKRNRQGLEIYYSINNPKVEGIIKLLNPIQ"
```

Para identificar seqüências específicas de todas as diferentes fontes (assim como as mudanças que podem ocorrer nas seqüências), a NCBI associou um identificador único para cada seqüência, denominado número 'GI'. Um novo número 'GI' é associado também para cada nova versão da seqüência. Esses identificadores aparecem no campo NID de um registro do GenBank, imediatamente após o campo ACCESSION. O número ACCESSION, em contraste, é associado para cada entrada no GenBank e não se modifica mesmo quando a seqüência ou uma anotação muda.

Existe um acordo entre os colaborativos bancos de seqüência de DNA para introduzir um terceiro identificador do qual irá agrupar ambos os números ACCESSION e 'GI'. GenBank irá apresentar esse identificador na linha chamada VERSION, que aparecerá abaixo da linha NID e será referenciado como 'Accession.version'. Por exemplo, uma entrada aparecendo no banco de dados pela primeira vez deverá ter o número de versão equivalente

ao número de acesso seguido de '.1' para refletir que esta é a primeira versão da seqüência desta entrada.

```
ACCESSION   Z000001
NID         g987655554321
VERSION     Z000001.1    GI: 987654321
```

A linha VERSION também irá mostrar o número 'GI'. Se a seqüência de nucleotídeos mudar, então a versão e o número 'GI' também mudarão, mas o número de acesso permanecerá o mesmo. Embora a linha NID carregue informação redundante, ela permanecerá na estrutura para garantir a compatibilidade com os programas existentes.

Seqüências de proteínas também terão um número identificador (no formato de três letras seguidas de cinco dígitos, isto é, ABC78912) que não mudarão, seguidas de um número de versão que será incrementado a cada subseqüente versão da seqüência. Isto irá aparecer como um qualificador para a característica CDS na tabela de características de uma entrada do GenBank.

```
/protein_id='ABC78912'
```

Traduções de proteínas correntemente recebem seu próprio número 'GI' que aparecem como um qualificador na característica CDS.

```
/db_xref='PID:g1234567'
```

Para finalizar a entrada, BASE COUNT irá identificar a quantidade das bases 'A', 'C', 'G' e 'T' na seqüência e ORIGIN poderá ser deixado em branco, ou classificado como não reportado, ou então poderá mostrar um trecho da seqüência, usualmente envolvendo o experimento, como mostrado a seguir:

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

- Recuperação dos dados do Genbank

O sistema *Entrez* é um sistema de banco de dados de recuperação integrado que acessa dados de seqüências de DNA e proteínas, referências MEDLINE relatadas, dados genéticos da divisão genética do Genbank, a taxinomia do NCBI e estruturas de proteínas do MMDB (*Molecular Modeling Database*). Os dados das seqüências de DNA e proteínas

foram integrados de uma variedade de fontes e incluem mais dados de seqüência do que o Genbank sozinho. Todo o conjunto de referências do MEDLINE se tornou acessível através do sistema PubMed que foi desenvolvido no NCBI e, além de permitir busca de texto no MEDLINE, também faz a ligação entre os textos de mais de 25 publicações que estão disponíveis na *Web*. *Entrez* está disponível na *Internet* [14] através da *Web* ou em uma versão cliente/servidor. A versão *Web*, incluindo a busca PubMed MEDLINE, é usada por mais de 40.000 usuários por dia. A versão cliente/servidor do *Entrez* opera com um programa cliente na máquina do usuário conectado através da *Internet* com um servidor localizado no NCBI.

Um dos usos mais freqüentes do Genbank é busca de seqüências por similaridade. O NCBI oferece a família BLAST de programas de buscas para executar uma procura rápida com rigorosos métodos estatísticos para julgar o significado das combinações. O acesso ao BLAST através da Web possui duas interfaces: uma versão 'básica' com parâmetros de busca default e uma opção 'avançada' que permite adaptação dos parâmetros. Existe uma ferramenta gráfica chamada Sequin que foi projetada para rápida análise e anotações de grande volume de dados de seqüências genéticas. Sequin é uma ferramenta desenvolvida pelo NCBI para desenvolver e atualizar entradas, tanto de seqüências simples quanto de seqüências mais complexas. Esta ferramenta está disponível no site no NCBI.

A versão completa do Genbank (atualizada a cada dois meses) ou as atualizações diárias estão disponíveis por FTP no [14]. O release completo no formato de arquivo texto está disponível como arquivos comprimidos no diretório 'genbank'. Um arquivo de atualizações cumulativas está contido no subdiretório 'daily' e um conjunto não-cumulativo de atualizações está no subdiretório 'daily-nc'. Desenvolvedores de software criam suas próprias interfaces e ferramentas de análise para os dados do Genbank e disponibilizam seus trabalhos no NCBI para ajudar o desenvolvimento de novas ferramentas. Softwares podem ser encontrados no diretório 'toolbox/ncbi_tools'.

Todas as informações contidas no Genbank estão armazenadas em um formato proprietário desse banco. Para possibilitar que um pesquisador utilize essas informações em conjunto com outras fontes, será necessário realizar um mapeamento entre os diferentes esquemas de representações de dados genéticos.

1.1.1.1 ACEDB

A proposta do ACEDB é um sistema de banco de dados que, além de armazenar os resultados do sequenciamento em larga escala e projetos de mapeamento, permite que todos os tipos de dados genéticos experimentais sejam mantidos e ligados aos mapas e seqüências

da maneira mais flexível possível, de modo que possa evoluir de acordo com a experiência adquirida [4].

Esse projeto começou com a obtenção da seqüência genética completa do *nematode C. elegans*. Existiam duas principais necessidades: primeiro, um sistema para manter os dados para propósitos internos; e segundo, um sistema para tornar os dados públicos. Isso levou ao desenvolvimento do programa de banco de dados ACEDB [4]. Em vez de ficar limitado ao dado específico, decidiu-se escrever um sistema geral de gerenciamento de banco de dados que permitiria que o esquema sofresse freqüente extensão e adaptação de forma fácil conforme o projeto fosse desenvolvido. Por essa razão, foi comparativamente fácil adaptar ACEDB para ser usado por outros projetos genéticos que trabalhassem com outros organismos. ACEDB está sendo usado também internamente por vários *sites*, por exemplo, para armazenar resultado do mapeamento físico dos projetos humanos e da *Drosophila*. Finalmente, está sendo usado como um dos pedaços centrais do *software* no projeto IGD, que planeja unir todos os dados do mapeamento genético humano em um banco de dados genéticos integrado. ACEDB está sendo usado como interface primária do IGD e como uma das alternativas de sistemas de armazenamento.

- Características gerais

De maneira geral, o programa é muito gráfico. Funciona usando um sistema de janelas e apresenta os dados em diferentes tipos de janelas de acordo com os diferentes tipos de mapas. Os mapas e outras janelas como um hipertexto, de forma que clicando sobre um objeto, serão apresentadas as informações detalhadas em uma janela específica. Por exemplo, clicando em um cromossomo mostra seu mapa genético. Clicando em um gene no mapa genético mostra informações textuais sobre o fenótipo do gene, etc.

ACEDB é um sistema orientado a objeto. Esse tipo de organização de dados é muito mais intuitivo que um baseado em tabelas relacionais. Devido a isso, permite uma entrada mais direta de biólogos trabalhando na construção e refinamento do esquema ou estrutura de dados.

Cada objeto é representado por um identificador único, seu nome, que é seguido por uma combinação de atributos organizados como uma árvore. Os nós da árvore são todos nomeados. Os galhos geralmente terminam em ponteiros para outros objetos ou dados. Há também a possibilidade de construir sub-objetos, expandindo recursivamente uma folha. Comentários podem ser adicionados livremente em qualquer ponto da árvore.

Os objetos são alocados em classes. Cada classe tem um modelo, especificando a extensão máxima dos galhos e os tipos de dados e classes de ponteiros permitidos em cada posição. Objetos individuais, que são instâncias de uma classe, geralmente têm apenas uma parte do padrão permitido pelo modelo. Essa abordagem tem três vantagens:

- Objetos pouco estudados, que são de longe os mais numerosos, ocupam pouco espaço em disco e em memória, que melhora significativamente a velocidade e a eficiência do programa.
- Quando é necessário estender um esquema, tudo que precisa ser feito é adicionar outro ramo no modelo. Nenhum dos objetos existentes contém aquele ramo, mas eles permanecem válidos porque não há requisito para os objetos conterem qualquer galho em particular do modelo.
- A habilidade de adicionar comentários pessoais que são ignorados pelos algoritmos de busca permite anotações flexíveis das fontes de dados sem afetar procedimentos internos de busca.

➢ Arquivos ACE

Embora os dados sejam armazenados internamente em formato binário de árvores, eles são normalmente informados através de arquivos ASCII conhecidos como arquivos ACE. Cada parágrafo nesses arquivos corresponde a um objeto e deve ser separado do próximo por uma ou mais linhas em branco. A primeira linha indica a classe e o nome do objeto a ser criado ou editado. As linhas seguintes começam com o nome do nó correspondente, seguido por dados numéricos ou textuais, ou nomes de outros objetos a serem apontados. Eles são interpretados de acordo com o modelo. Palavras-chave como '–D' ou '–R' especificam ações a serem tomadas, com a ação default sendo adicionar o dado no banco de dados. Assim como em C++, '//' indica um comentário no arquivo.

Exemplo [4]:

```
// definição de uma seqüência
Sequence ACT3
Title ``C. elegans actin gene (3)"
Library EMBL CEACT3 X16798


// o DNA correspondente
DNA ACT3
aagagagacatcctcccgctcccttcccacacccacttgctcttttctat
tgaccacacattatgaagataaccatgttactaatcaaattcgtgttctt
ttccaatttctttttc

// Aqui o nome zk643 é mudado (caso exista)
-R Sequence zk643 ZK643          // R for ``rename"
```

```
// aqui é mudado um dos autores
Paper Nurture:7:234-242
-D Author ``Kimble JE"      // deletion of Kimble
Author ``Ahringer JA"       // addition of Ahringer
```

Devido aos objetos terem um identificador ASCII único, [classe:nome], esses comandos de edição são bem-definidos e se referem a objetos precisos. Se o objeto não é conhecido ainda, ele é criado, caso contrário é modificado. Se a operação de apagar ou renomear não acha nada para apagar ou renomear, pula para próxima operação. Se uma instrução não faz nenhum sentido de acordo com o modelo (por exemplo, se referindo a um nó desconhecido), o usuário é avisado e o parágrafo é pulado. Essas propriedades em conjunto permitem a leitura repetida do mesmo arquivo sem alterar o conteúdo do banco de dados e a transferência de informações entre bancos de dados que podem não combinar exatamente, algo que é muito difícil com os sistemas de bancos de dados tradicionais. Além disso, eles também permitem a transferência de dados significativos comuns entre sistemas cujo esquema difere.

Assim como a leitura de dados, também pode ser feita a exportação de um conjunto de objetos no formato de arquivo ACE. Um programa externo, ACEDIFF, recebe como entrada dois arquivos ACE e gera um terceiro que terá o efeito de transformar o dado contido no primeiro arquivo no dado contido no segundo arquivo. Esse programa pode ser usado para gerar arquivos de atualização para cópias remotas de um banco de dados central e de fato, esse é o procedimento adotado para distribuir o banco de dados genético do *nematode C. elegans*. Também há outras facilidades dentro do ACEDB para certos tipos de saídas especializadas de dados, como DNA no formato FASTA.

Assim como ocorre no Genbank, as informações do ACEDB estão armazenadas em um formato proprietário, necessitando realizar um mapeamento entre os diferentes esquemas de representações de dados genéticos para possibilitar a utilização dessas informações em conjunto com outras fontes.

2.2.2.2. Swiss-Prot

Swiss-Prot [5] é um banco de dados de seqüências de proteínas com anotações. Começou em 1986 e foi mantido desde 1987 com colaboração do grupo de Amos Bairoch, primeiramente no Departamento de Medicina Bioquímica da Universidade de Genebra e agora pelo Instituto Suíço de Bioinformática (SIB) e pela Biblioteca de Dados do EMBL. O conhecimento de proteínas do Swiss-Prot consiste de entradas de seqüências. Entradas de seqüências são compostas de diferentes tipos de linhas, cada qual com seu próprio formato. Com propósito de padronização o formato do Swiss-Prot segue o mais próximo possível do EMBL *Nucleodite Sequence Database*.

Swiss-Prot se distingue dos bancos de dados de proteínas por quatro fatores:

a) Anotações

No Swiss-Prot, como em muitos bancos de dados de seqüências, duas classes de dados podem ser encontradas: o dado em si e as anotações.

Para cada entrada de seqüência o dado consiste de:

- Os dados da seqüência;
- As referências bibliográficas;
- O dado taxinômico (descrição da fonte biológica da proteína).

A anotação consiste na descrição dos seguintes itens:

- Funções da proteína;
- Modificações *postranslational*;
- *Sites* e domínios;
- Estruturas secundárias;
- Estruturas quaternárias;
- Similaridades com outras proteínas;
- Doenças associadas com qualquer número de deficiências na proteína;
- Conflitos na seqüência, variantes, etc.

É armazenada no Swiss-Prot a maior quantidade possível de anotações. Para obter essa informação, além das publicações que reportam novas seqüências de dados, artigos de revisão para atualizar periodicamente as anotações de famílias ou grupos de proteínas. São também utilizados *experts* externos, que são recrutados para enviar seus comentários e atualizações sobre grupos específicos de proteínas.

As anotações no Swiss-Prot são principalmente encontradas em linhas de comentário (CC – *comment*), na tabela de características (FT – *feature table*) e em linhas

de palavras-chave (KW – *keyword*). A maioria dos comentários é classificada por 'tópicos'. Essa forma permite fácil recuperação de categorias específicas de dados do banco de dados.

b) Redundância mínima

Muitos bancos de dados de seqüências contêm, para uma dada seqüência de proteínas, entradas separadas que correspondem a diferentes relatórios. No Swiss-Prot é feita uma tentativa de conciliar todos esses dados de forma a minimizar a redundância do banco de dados. Se existem conflitos entre vários relatórios de seqüências, eles são indicados na tabela de características da entrada correspondente.

c) Integração com outros bancos de dados

É importante prover aos usuários dos bancos de dados biomoleculares um certo grau de integração entre os três tipos de bancos de dados relacionados a seqüências (seqüências de ácidos nucléicos, seqüências de proteínas e estruturas ternárias de proteínas) assim como com coleções de dados especializados. Swiss-Prot tem normalmente referência cruzada com cerca de 45 bancos de dados diferentes. As referências cruzadas estão na forma de ponteiros para informações relacionadas às entradas do Swiss-Prot e são encontradas em coleções de dados diferentes do Swiss-Prot. Essa rede extensa de referências cruzadas permite ao Swiss-Prot representar um papel importante como um ponto focal de interconectividade entre bancos de dados biomoleculares.

d) Documentação

Swiss-Prot é distribuído com um grande número de arquivos de índice e de documentação especializada. Alguns desses arquivos estão disponíveis há muito tempo, outros foram incluídos recentemente e continuamente novos arquivos são incluídos.

➢ Estrutura geral do banco de dados

O Swiss-Prot é um banco de dados composto de entradas de seqüências de proteínas. Cada entrada corresponde a uma única seqüência contígua tal como foi incluída no banco ou reportada na literatura. Em alguns casos, as entradas foram montadas de várias publicações sobre regiões sobrepostas das seqüências.

Referências a posições dentro das seqüências são feitas usando numeração seqüencial, começando com 1.

Para fazer o dado ficar disponível aos usuários o mais rápido possível depois da publicação, Swiss-Prot está agora disponível com um suplemento TrEMBL, onde as entradas são liberadas antes de todos os seus detalhes serem finalizados. Para distinguir entre entradas com anotações e aquelas em TrEMBL, a 'classe' de cada entrada é indicada na primeira linha (ID) da entrada. As duas classes definidas são:

- STANDARD: dados que estão completos e de acordo com o nível do banco de dados Swiss-Prot.
- PRELIMINARY: entradas de seqüências que ainda não foram anotadas pelo pessoal do Swiss-Prot. Essas entradas são encontradas exclusivamente em TrEMBL.

As entradas no Swiss-Prot são estruturadas de forma a serem utilizadas tanto por leitores humanos quanto por programas de computador. As explicações, descrições, classificações e outros comentários estão em inglês coloquial. Símbolos familiares a bioquímicos, químicos de proteínas e biólogos moleculares são usados sempre que possível.

Cada entrada de seqüência é composta de linhas. Diferentes tipos de linha, cada uma no seu próprio formato, são usados para gravar os vários dados para fazer a entrada. Cada linha começa com um código de linha de dois caracteres que indica o tipo de dado contido naquela linha.

Exemplo (parte de um registro) [5]:

```
ID  GRAA_HUMAN     STANDARD;      PRT;   262 AA.
AC  P12544;
DT  01-OCT-1989 (Rel. 12, Created)
DT  01-OCT-1989 (Rel. 12, Last sequence update)
DT  16-OCT-2001 (Rel. 40, Last annotation update)
DE  Granzyme A precursor (EC 3.4.21.78) (Cytotoxic T-lymphocyte proteinase
DE  1) (Hanukkah factor) (H factor) (HF) (Granzyme 1) (CTL tryptase)
DE  (Fragmentin 1).
GN  GZMA OR CTLA3 OR HFSP.
OS  Homo sapiens (Human).
OC  Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
OC  Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
OX  NCBI_TaxID=9606;
RN  [1]
RP  SEQUENCE FROM N.A.
RC  TISSUE=T-cell;
RX  MEDLINE=88125000; PubMed=3257574;
(... a parte do meio foi cortada)
FT  DISULFID   208   234      BY SIMILARITY.
FT  CARBOHYD   170   170      N-LINKED (GLCNAC...) (POTENTIAL).
SQ  SEQUENCE   262 AA;  28968 MW;  DA87363A0D92BAF4 CRC64;
    MRNSYRFLAS SLSVVVSLLL IPEDVCEKII GGNEVTPHSR PYMVLLSLDR KTICAGALIA
    KDWVLTAAHC NLNKRSQVIL GAHSITREEP TKQIMLVKKE FPYPCYDPAT REGDLKLLQL
    TEKAKINKYV TILHLPKKGD DVKPGTMCQV AGWGRTHNSA SWSDTLREVN ITIIDRKVCN
    DRNHYNFNPV IGMNMVCAGS LRGGRDSCNG DSGSPLLCEG VFRGVTSFGL ENKCGDPRGP
    GVYILLSKKH LNWIIMTIKG AV
//
```

Os atuais tipos e códigos de linha e a ordem em que eles aparecem em uma entrada estão descritos na tabela 2-1.

Tabela 2-1: Estrutura de uma entrada Swiss-Prot [5]

Line code	Content	Occurrence in an entry
ID	Identification	Once; starts the entry
AC	Accession number(s)	Once or more
DT	Date	Three times
DE	Description	Once or more
GN	Gene name(s)	Optional
OS	Organism species	Once or more
OG	Organelle	Optional
OC	Organism classification	Once or more
OX	Taxonomy cross-reference(s)	Once or more
RN	Reference number	Once or more
RP	Reference position	Once or more
RC	Reference comment(s)	Optional
RX	Reference cross-reference(s)	Optional
RA	Reference authors	Once or more
RT	Reference title	Optional
RL	Reference location	Once or more
CC	Comments or notes	Optional
DR	Database cross-references	Optional
KW	Keywords	Optional
FT	Feature table data	Optional
SQ	Sequence header	Once
	(blanks) sequence data	Once or more
//	Termination line	Once; ends the entry

Como mostrado na tabela 2-1, alguns tipos de linhas são encontrados em todas as entradas, outros são opcionais. Alguns tipos de linhas ocorrem muitas vezes em uma única entrada. Cada entrada deve começar com uma identificação de linha (ID) e terminar com um terminador de linha (//).

O código de tipo de linha de que começa cada linha é sempre seguido de três caracteres brancos, então realmente, a informação só começa no sexto caractere.

2.2.2.3. Protein Data Bank

O Protein Data Bank (PDB) [27, 38] foi criado no Brookhaven National Laboratories (BNL) em 1971 como um arquivamento para estruturas macromoleculares em formatos de cristais. No início, o arquivamento continha sete estruturas, mas a cada ano várias outras estruturas eram armazenadas. Na década de 80, o número de estruturas armazenadas aumentou drasticamente.

O uso inicial do PBD estava limitado a um pequeno grupo de pessoas envolvidas em pesquisas estruturais. Atualmente, a utilização do PBD se estendeu a diversos grupos de pesquisa em biologia e química, cientistas da computação, educadores e estudantes de todos os níveis. O imenso aumento de entrada de dados e da necessidade de entendê-los completamente, demanda novos métodos de coletar, organizar e distribuir os dados.

Em outubro de 1998, o gerenciamento do PDB se tornou responsabilidade do Research Collaboratory for Structural Bioinformatics (RCSB). Em termos gerais, a visão do RCSB é criar uma pesquisa baseada na mais moderna tecnologia que facilita o uso e análise de dados estruturados e permite um maior aproveitamento das pesquisas biológicas.
O principal objetivo do PDB é fazer o arquivamento o mais consistente e sem erros possível. Todos os dados depositados são revisados cuidadosamente antes de ficarem disponíveis. Tabelas de características são geradas a partir do processamento de dados interno do banco e depois são checadas. Erros encontrados pelos autores ou usuários do PDB após a disponibilização são endereçados o mais rápido possível.

Um dos problemas mais complicados que o PDB enfrenta é que os arquivos legados não são uniformes. Historicamente, dados legados estão em vários diferentes formatos de PDB e variações. A introdução da capacidade de consultas avançadas do PDB torna crítica a aceleração do processo de uniformidade desses dados. Além dos esforços do PDB para remediar antigas entradas de dados, o European Bioinformatics Institute (EBI) também corrige muitos dos arquivos de PDB como parte do projeto de limpeza. A tarefa de integração de todas essas correções feitas dos dois lados é muito grande e é essencial que haja um formato bem definido de troca para fazer isso. O formato mmCIF é utilizado para esse propósito.

- Diferenças entre PDB e mmCIF

O formato de dados representado por mmCIF é significantemente diferente do formato PDB [34]. Existem ferramentas de conversão de arquivos de dados PDB em mmCIF que possuem um bom resultado. Entretanto, as dificuldades surgem devido à forma ambígua e inconsistente que as informações estão armazenadas dentro do arquivo PDB. É importante ressaltar que durante a conversão de volta para PDB geralmente ocorre perda de informação. Enquanto o formato mmCIF representa um claro avanço sobre o PDB, sua adoção tem sido lenta. Uma das razões é o investimento existente no formato PDB com respeito a ferramentas de visualização e análise. Os mesmos desafios serão enfrentados por qualquer formato que tente substituir o PDB ou o mmCIF. Para superar esse desafio, um novo formato deve ser tão

interessante que motive os desenvolvedores e vendedores de ferramentas a suportar o novo formato, ou então, prover tradutores robustos do novo formato para um formato já bastante conhecido e vice-versa. A segunda estratégia, mesmo requerendo um montante de trabalho bem maior, parece ser a mais viável. Ferramentas para manipulação de dados representados em documentos XML (independente dos tipos de dados codificados ou do domínio específico) estão largamente disponíveis. Além disso, o suporte em diferentes operações em dados baseados em XML é largamente suportado por produtos comerciais e projetos de ferramentas de código público. A coleção de ferramentas genéricas de XML, combinadas com a evolução do XML como um padrão de troca de dados, tornam o argumento de formatos de dados de proteínas baseados em XML mais atraentes que alternativas que não são baseadas em XML, como o mmCIF. Não é surpresa que um grande número desses formatos exista, embora nenhum deles tenha ganhado forma suficiente para substituir os formatos não baseados em XML.

- Arquitetura do banco

Em reconhecimento ao fato de que nenhuma arquitetura pode expressar completa e eficientemente o conteúdo das informações contidas no PDB, um sistema integrado de bancos de dados heterogêneos foi criado para armazenar e organizar os dados estruturados. Existem cinco principais componentes que podem ser visualizados na figura 2-2:

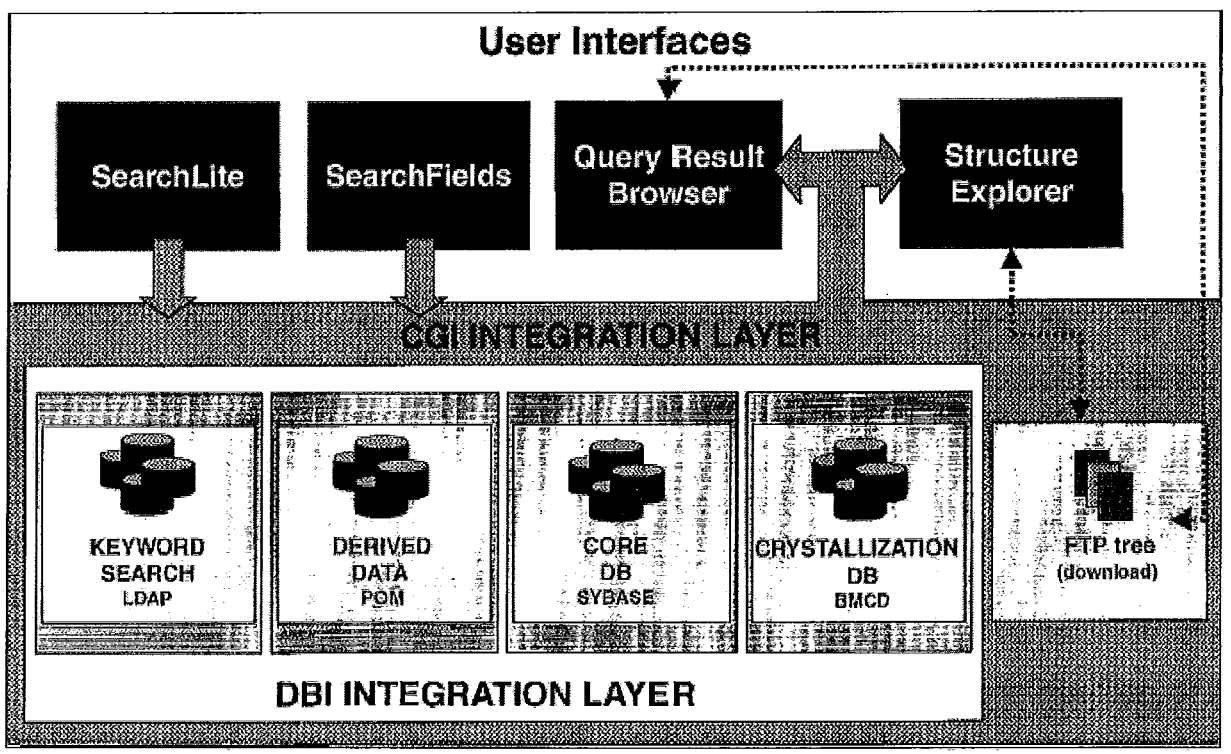


Figura 2-2: Interface de consulta integrada do PDB [38]

- O principal banco de dados relacional é gerenciado pelo Sybase e provê o armazenamento físico central dos dados primários experimentais e coordenadas. Esse banco contém todas as informações depositadas na forma de tabelas que podem ser acessadas através de várias estruturas.
- Os arquivos finais (em formatos PDB e mmCIF) e os dicionários de dados são os dados arquivados e estão presentes como arquivos ASCII no arquivamento ftp.
- Os bancos de dados baseados em POM (Property Object Model) consistem de objetos indexados contendo propriedades nativas (por ex., coordenadas atômicas) ou derivadas (por ex., associações secundárias calculadas). Propriedades que requerem um tempo significativo de computação são pré-calculadas quando o banco de dados é incrementado, salvando assim um tempo considerável de acesso do usuário.
- O Biological Macromolecule Crystallization Database (BMCD) é organizado com um banco relacional dentro do Sysbase e contém três categorias gerais: dados macromoleculares, cristais e sumários.
- O servidor Netscape LDAP é usado para indexar o conteúdo textual do PBD em um formato estruturado e prover suporte para buscas por palavras-chave.

É fundamental que toda essa arquitetura seja transparente para o usuário. Na implementação atual, a comunicação entre os bancos de dados foi conseguida usando o Common Gateway Interface (CGI). Uma interface integrada na Web dispara a consulta ao banco apropriado que executa a consulta. Cada banco de dados retorna os identificadores do PDB que satisfazem a consulta e o programa CGI integra os resultados. Consultas complexas são feitas repetindo-se o processo e tendo o programa de interface executado as operações booleanas apropriadas na coleção de resultados da consulta. Várias opções de saída estão disponíveis para o uso com a lista final de estruturas selecionadas.

A solução CGI (e no futuro uma solução CORBA – Common Obejct Request Broker Architecture) permite que outros bancos de dados se integrem nesse sistema, por exemplo, dados estendidos de diferentes famílias de proteínas.

2.2.2.4. Protein Extensible Markup Language (PROXIML)

PROXIML [33] é resultado de uma pesquisa da Universidade da Califórnia sobre os problemas dos diferentes formatos de dados de proteínas, como PDB e mmCIF [34], e a necessidade de se ter uma maneira mais autodescritiva e de fácil leitura para troca desses dados. XML é uma solução ideal para esse problema em particular. Entretanto, alguns esforços baseados em XML (como BIOML [35] e ProML [36]) utilizam DTD para descrever

e validar a estrutura dos dados. A utilização do DTD impõe uma série de restrições sobre a estrutura e a habilidade de validação de um documento XML. Uma alternativa é a utilização do XML Schema que supera essas limitações. O PROXIML é um esquema descrito em XML Schema que propõe uma maneira genérica de descrever os dados sobre proteínas. O PROXIML é uma extensão do esquema CML [33] que utiliza XML Schema para descrever dados químicos.

- Comparação entre diferentes formatos existentes

Uma comparação em alto nível dos formatos mencionados anteriormente é apresentada na tabela 2-3 [33]. O formato PROXIML tem como propósito criar um esquema juntando as melhores características de cada um.

Na tabela 2-3, os diversos formatos são analisados segundo as seguintes características: se os formatos são baseados em XML; se os formatos possuem um arquivo de metadados que os descrevem; se os formatos são autodescritivos; se os formatos estão preparados para tratar dados de seqüências, dados sobre as famílias genéticas e dados de anotações; se os formatos são estruturados e se os formatos possuem ferramentas de conversão e visualização.

Tabela 2-3: Comparação dos formatos existentes de dados sobre proteínas [33]

	PDB	mmCIF	BIOML	ProML	CML
XML Based?	No	No	Yes	Yes	Yes
Metadata	None	DDL	DTD	DTD	Schema
Self-describing?	No	Yes	Yes	Yes	Yes
Sequence data?	Yes	Yes	No	Yes	Limited
Family data?	No	No	Yes	Good	No
Annotation?	Yes	Yes	Good	Yes	Limited
Structure?	Yes	Yes	No	Yes	Good
Conversion tools?	Many	Some	Few	Few	Some
Visualization tools?	Many	Some	Few	No	Good

- Implementação PROXIML

A versão inicial do formato PROXIML é um híbrido de três formatos diferentes: BIOML, ProML e CML. O coração do formato do documento é uma estrutura modificada baseada no elemento *<molecule>* do CML. Esse elemento está agrupado dentro da porção *<data>* de cada elemento *<protein>* incluído do documento PROXIML. Cada proteína

descrita em PROXIML possui três elementos de alto nível, dois deles derivados do formato ProXML (*<identity>* e *<data>*) e um deles derivado do BIOML (*<annotation>*) que encapsula a maioria da estrutura do documento BIOML. A implementação corrente do PROXIML requer um trabalho significante para ligar de forma apropriada os elementos de anotações com os outros elementos associados nas porções *<identity>* e *<data>* do elemento *<protein>*, e para minimizar a redundância e a possibilidade de valores de dados conflitantes entre partes do documento. O esquema completo do PROXIML e exemplos estão disponíveis em [33].

A adoção desse formato vai depender principalmente da disponibilidade de ferramentas (como ferramentas de visualização e conversão) que suportem esse novo formato.

Novamente, temos informações armazenadas em um formato proprietário tornando necessário realizar um mapeamento entre os diferentes esquemas de representações de dados genéticos para possibilitar a utilização dessas informações em conjunto com outras fontes.

2.2.2.5. GAME e AGAVE

Tanto GAME quanto AGAVE são propostas de formatos genéricos e unificados para dados genéticos baseados em XML [39, 40].

GAME (*Genome Annotation Markup Elements*) é uma tentativa de criar um DTD [8] para as anotações de seqüências biológicas. É um formato aberto que pode ser usado para representar informações sobre regiões específicas de uma seqüência, permitindo a diferenciação de diversos tipos de características, tais como aquelas geradas por um programa de análise e outras geradas por profissionais humanos. A primeira versão largamente utilizada desse formato foi criada no BDGP (*Berkeley Drosophila Genome Project*) por Suzanna Lewis e Erwin Frise [54].

AGAVE (*Architeture for Genomic Annotation, Visualization and Exchange*) também é um formato aberto baseado em XML que oferece para a comunidade científica uma forma de compartilhar suas anotações genéticas. Descreve o relacionamento entre seqüências de fragmentos contíguos e seus componentes, a orientação e a ordem dos fragmentos e as características associadas a cada um deles. O principal objetivo do AGAVE é prover uma linguagem de marcação para anotações genéticas que seja compreensível, extensível, aberta e de fácil leitura. É compreensível porque pode representar todos os dados biológicos relevantes em bancos de dados públicos como Genbank, incluindo o formato da localização da seqüência completa. É extensível porque usa elementos genéricos para resultados

computacionais que podem facilmente ser usados para capturar resultados de novos algoritmos de anotações genéticas. Devido ao formato AGAVE ser baseado em XML, é facilmente utilizado como um padrão aberto para armazenamento e troca de dados. Para conseguir uma fácil leitura, AGAVE foca em dados biológicos e usa terminologias biológicas padrões com o mínimo de abreviação.

AGAVE e GAME têm elementos similares para representar seqüências e anotações das seqüências. Ao contrário do AGAVE, GAME não possui a habilidade de representar características recursivas e montagens contíguas.

Existem ferramentas disponíveis para o AGAVE e GAME de visualização e conversão para outros formatos [40] que realizam conversões entre as representações de dados genéticos previamente definidas de forma automática. Essas ferramentas não permitem adaptações do usuário, sendo necessário construir novas ferramentas para suportar outros tipos de representação.

2.2.2.6. Conclusão

Durante essa seção vimos os principais bancos de dados e tipos de arquivos genéticos utilizados atualmente e seus formatos, seus objetivos e como as características inerentes aos dados genéticos são tratadas.

Percebemos nitidamente a diversidade de tratamento e análise de tipos de dados genéticos que são muito similares entre si. Toda essa diversidade de opções acaba dificultando o dia-a-dia dos pesquisadores da área genética. Essa dificuldade se torna mais complicada quando consideramos que a integração entre os diversos grupos de pesquisa é essencial, já que as pesquisas genéticas geralmente são correlacionadas e não existe um padrão único de representação de dados genéticos.

Para tentar resolver ou diminuir esses desafios encontrados pelos pesquisadores, foram propostos vários trabalhos. No próximo capítulo veremos alguns desses trabalhos e como nossa proposta se encaixa nesse cenário.

CAPÍTULO 3. GERÊNCIA DE FONTES HETEROGÊNEAS DE DADOS E PROGRAMAS GENÉTICOS

Nesse capítulo descrevemos os principais projetos que trabalham com gerência de dados e programas na área de bioinformática. Em especial são apresentadas as arquiteturas Bio-Axis [51] e SRMW [53] que motivaram o desenvolvimento desta dissertação.

Projetos sobre genomas moleculares, através do sequenciamento genético, têm produzido enormes coleções de dados de DNA de diversos organismos. Uma preocupação importante dessa área de pesquisa é como lidar com os dados genéticos, de forma a identificar suas funções. Além disso, é vital fazer comparações entre os dados genéticos dos indivíduos que podem ser da mesma espécie ou não. Muitos grupos têm trabalhado no desenvolvimento de ferramentas para prover integração, estruturação, comparação e anotação das seqüências genéticas e suas informações relacionadas.

Os projetos genoma normalmente desenvolvem suas próprias interfaces para os sistemas, e diferem na forma de apresentar seus dados e resultados da manipulação de dados. Por exemplo, o *AceDB* oferece uma interface gráfica poderosa que permite ao usuário visualizar o mapeamento de cromossomos em detalhes.

Vários grupos de pesquisa estão preocupados com os problemas relacionados à integração das diversas fontes de dados e programas genéticos. Basicamente podemos dividir esses grupos de pesquisa de acordo com suas propostas para resolver o problema de integração. Temos os grupos de pesquisa que propõem um modelo único global de representação de dados genéticos. Nesse caso todos as fontes de dados seriam convertidas para esse modelo global e os programas seriam adaptados para funcionar com esse mesmo modelo. Seguindo essa linha, temos o trabalho proposto por Paton, "Modelagem Conceitual de Informações Genéticas" [41] e o projeto GIMS [64, 65]. Com a adoção de um modelo único global, o acesso aos dados fica mais rápido e pode ser otimizado de modo a agilizar as análises e o processo de armazenamento e recuperação. Além disso, as informações e os programas são facilmente reaproveitados para novas análises. O grande problema com esse tipo de abordagem é conseguir um consenso com a comunidade científica de qual modelo seria indiscutivelmente mais adequado para ser adotado como um modelo único global. Além disso, deve-se levar em conta o esforço necessário para se converter as fontes de dados atuais para esse novo modelo global.

Outra linha de pesquisa se propõe a criar um ambiente integrado para que os biólogos possam trabalhar com as diversas fontes de dados em seus formatos originais, realizando o *workflow* dos diversos programas necessários para a continuidade das suas pesquisas.

Acreditamos que essa linha de pesquisa seja a mais adequada para a área genética. Atualmente já existe uma imensa quantidade de dados armazenados das mais diversas formas e os pesquisadores já estão acostumados a trabalhar com seus dados e *workflows* em um determinado formato. Em vez de forçar a todos se adaptarem a um novo modelo global, acreditamos que a melhor solução é a criação de um ambiente que permita a convivência e a integração das diversas fontes de dados e programas científicos em seus formatos atuais.

Seguindo essa linha, temos ainda algumas divisões nas propostas oferecidas. Podemos citar o projeto MyGrid [42] que propõe a utilização de uma nova plataforma de *middleware* para suportar o compartilhamento coordenado de fontes de dados e soluções na escala global para aplicações com intensa quantidade de dados e computação, baseado em modelos de ontologias.

Outro projeto nessa linha é o projeto DataFoundry desenvolvido pelo Lawrence Livermore National Laboratory (LLNL) [44] que propõe uma infra-estrutura baseada em metadados para suportar uma arquitetura de *data warehouse* (DW) com mediadores, com o objetivo de melhorar a interação dos cientistas com seus dados. Cada mediador se encarrega de um tipo de integração.

Também nessa linha, temos os projetos que propõem a utilização de um *framework* que possa criar um ambiente integrado para trabalhar com os diversos dados e programas científicos. O projeto "Bio-AXS" [6, 51] propõe uma arquitetura de *framework* para integração de fontes de dados e aplicações da biologia molecular.

Com uma proposta voltada para a composição de programas, foi desenvolvido o projeto "Gerenciamento de Recursos Científicos baseado em Web Services" [37, 45] que propõe uma arquitetura para o gerenciamento de recursos científicos distribuídos entre bases de dados e metadados.

Existem grupos que analisaram os desafios característicos da área de bioinformática e quais seriam as principais linhas de ação. Podemos citar o *workshop* XEWA-00 [18] que teve como objetivo reunir membros da comunidade de bioinformática para determinar se o uso de XML poderia simplificar o acesso às largas fontes de dados genéticas e heterogêneas que estão distribuídas pela *Web*.

Outros grupos de pesquisa propõem a estudar a utilização de ontologias na área de bioinformática. Essas ontologias podem ser utilizadas para facilitar o mapeamento entre as diversas fontes de dados e programas científicos. As ontologias podem ser usadas em mapeamentos tanto nas propostas de utilização de um modelo único global quanto em arquiteturas de integração das diversas fontes distribuídas disponíveis. O trabalho "Avaliação

de Linguagens para Troca de Ontologias em Bioinformática" [48] avalia diversas linguagens para troca de ontologias na área de bioinformática e recomenda uma ou mais linguagens para serem usadas.

Na tabela 3-1 apresentamos um resumo comparativo dos trabalhos relacionados que mencionamos, onde a primeira abordagem segue a linha de um modelo único global de representação de dados genéticos, e a segunda abordagem propõe criar um ambiente integrado para que os biólogos possam trabalhar com as diversas fontes de dados em seus formatos originais.

Tabela 3-1: Comparação dos trabalhos relacionados

Projetos	1ª abordagem	2ª abordagem	Principais Características
GIMS [64, 65]	X		DW para armazenamento e análise de seqüências genéticas e dados funcionais
MyGrid [42]	X		Middleware com ontologias para suportar compartilhamento coordenado de recursos científicos
DataFoundry [44]	X		DW com mediadores para representar globalmente fontes de dados
Bio-AXS [6, 51]		X	Framework OO para integração de fontes de dados e aplicações da biologia molecular
SRMW [37, 45]		X	Gerenciamento de recursos científicos baseado em Web Services
GAME [39]	X		Proposta de representação de dados genéticos em XML
AGAVE[40]	X		Proposta de representação de dados genéticos em XML
PROXIML [33]	X		Proposta de representação de dados genéticos em XML

Nossa proposta de trabalho se encaixa dentro da linha de pesquisa de ambientes integrados para trabalhar com as diversas fontes de dados em seus formatos originais e seus *workflows*. Nessas arquiteturas é necessário que o bioinformata ou biólogo informe as regras de negócio que possibilitam a integração desejada. Na maioria das vezes, o mapeamento dessas informações não é trivial e realmente depende da interação com o biólogo. Nossa proposta é um módulo que dê suporte ao bioinformata durante o processo de mapeamento das informações genéticas, facilitando a integração de diferentes fontes de dados e/ou entradas e saídas de programas. Esse módulo pode ser utilizado em conjunto com outras arquiteturas para disponibilizar aos biólogos um ambiente integrado de armazenamento e análise dos dados genéticos independentemente dos formatos em que estão originalmente disponíveis.

Durante as próximas seções vamos descrever com mais detalhes os trabalhos mais relacionados a essa dissertação. No capítulo 3 veremos com detalhes o módulo de suporte ao mapeamento que desenvolvemos.

3.1. Modelagem Conceitual de Informações Genéticas

O objetivo desse trabalho [41] é prover modelos conceituais claros para dados genéticos que possam ser usados diretamente para implementações subseqüentes. Separando os modelos de informação da descrição de um sistema, é esperado que itens importantes relacionados ao modo como o dado pode ser descrito se tornem explícitos, beneficiando os desenvolvedores de futuros sistemas para dados genéticos.

Esse projeto apresenta modelos conceituais para descrever tanto seqüências de genoma quanto conjuntos de dados funcionais relacionados. Os modelos produzidos servem de ponto inicial para uma atividade de implementação, na qual os diagramas de classes da UML são mapeados para um objeto do esquema do banco de dados. Entretanto, como representações conceituais de conjuntos de dados, esses modelos podem ser igualmente mapeados para diferentes plataformas de implementação, tais como modelos de dados alternativos, modelos de objetos para dados distribuídos (por ex. CORBA) ou dados temporários (por ex. JAVA).

Projetos de sequenciamento genético e funcional estão tornando disponíveis novas fontes de informação que vão motivar o desenvolvimento de uma nova geração de bioinformática. Entretanto, está claro que o armazenamento de informações e os desafios de análise são extremamente grandes. Uma análise efetiva dos genomas requer alto desempenho de acesso a uma diversa coleção de informações de pesquisa, que tipicamente estão espelhadas em diversos sites e em diversos formatos. Essas informações geralmente estão em um formato que permite a navegação e salvamento dos dados, mas não necessariamente analisado efetivamente em conjunto com outras fontes de informação.

Vários pesquisadores investigam ferramentas para consultas distribuídas sobre fontes de informações sobre seqüências genéticas. Os autores desse trabalho acreditam que a proposta de *warehousing* será interessante também nessa área. A diferença principal em genética é que análises típicas tendem a ser mais complexas que outras áreas. Em genética, os valores são adicionados através das associações mais próximas de dados específicos pesquisados. Dessa forma os genomas podem ser comparados uns com os outros, e as informações dos experimentos podem ser facilmente associadas e mostradas.

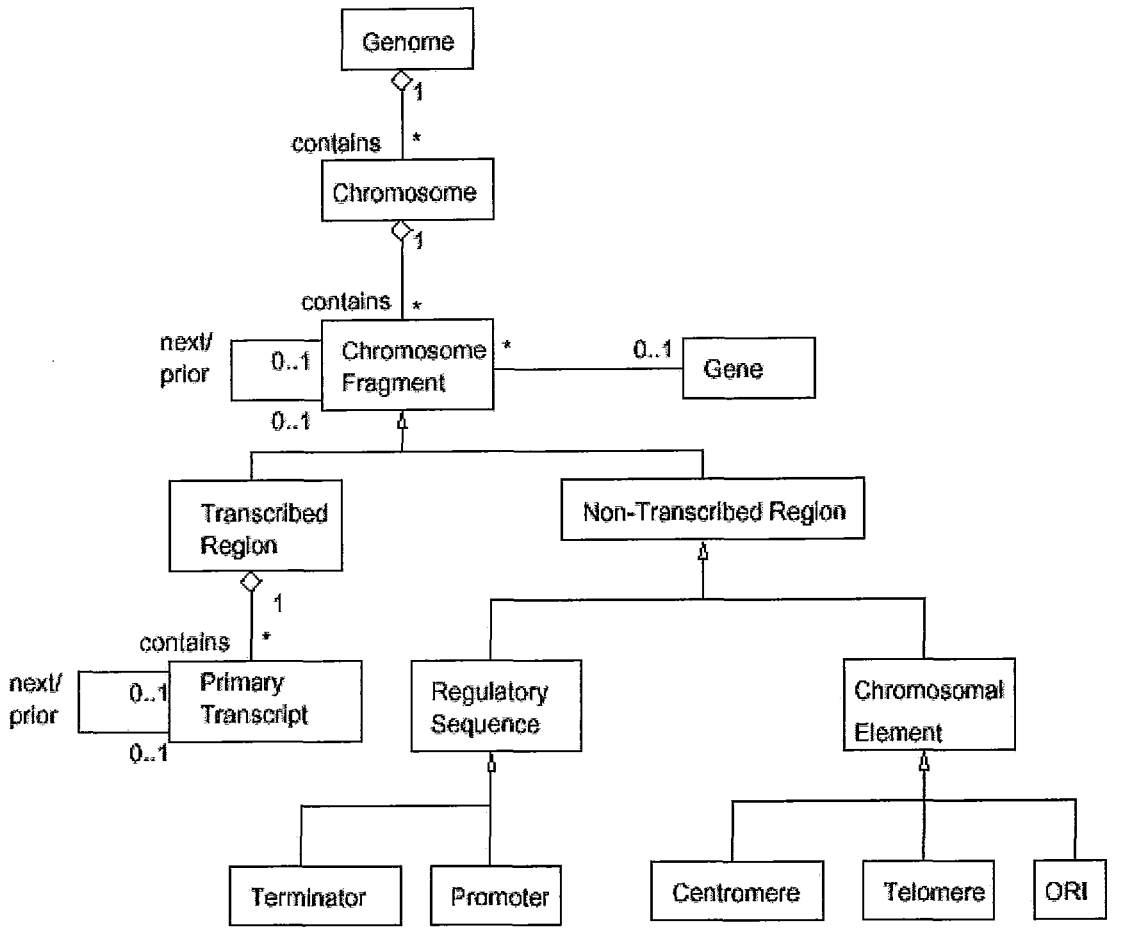


Figura 3-2: Diagrama do esquema básico para dados genéticos [41]

Na figura 3-2 podemos ver a modelagem básica para os dados genéticos mostrada em [41].

Esse trabalho procura prover uma fundação para construção de outros trabalhos, porque para que esses modelos sejam aceitos e largamente utilizados, será preciso implementá-los, e técnicas de busca, análise e visualização precisam ser desenvolvidas para uso desses modelos.

Se esse modelo fosse adotado como um modelo único por todos os pesquisadores, esta poderia ser uma solução para o problema de integração, mas o problema é exatamente convencer a comunidade científica a adotar essa modelagem. Para utilizar os modelos propostos, seria preciso desenvolver ferramentas para trabalhar com esses modelos.

Além de adotar essa modelagem para novas pesquisas, seria preciso converter os dados já existentes. A ferramenta que desenvolvemos serviria para dar o suporte necessário durante o mapeamento das informações e programas existentes para a nova modelagem.

3.2. Projeto MyGrid

A plataforma computacional Grid [42, 55, 56, 57, 58] é uma plataforma de *middleware* para suportar compartilhamento coordenado de fontes de dados e soluções na escala global para aplicações com intensa quantidade de dados e computação. É um projeto pioneiro que é elogiado por vários outros. O nome surgiu da analogia com grade de energia elétrica: fontes de dados e computacionais serão distribuídos pela Internet de forma igual, transparente e dinâmica como e quando forem precisos, tal como a eletricidade. A motivação geral é compartilhar poder computacional.

Esse *middleware* deve permitir a construção de novas capacidades a partir de serviços distribuídos de forma dinâmica e transparente. Para isso, é desejável ser capaz de reusar componentes existentes e fontes de informações, e montar e coordenar esses componentes de uma maneira flexível.

As configurações de serviço da Grid são:

- Dinâmico e volátil: serviços (bancos de dados, sensores, servidores) participando de uma análise complexa podem ser ligados e desligados conforme eles se tornem disponíveis ou indisponíveis;
- *Ad-hoc*: serviços não possuem uma localização central e não existem relacionamentos de confiança;
- Grande: centenas de serviços podem ser orquestrados a qualquer momento;
- Vida longa: uma simulação pode levar semanas.

Os serviços da Grid são organizados em quatro camadas:

- Fábrica: segurança, transporte de dados, certificação, acesso remoto, monitoramento de rede, autenticação;
- Base: programação de recursos, acesso de dados, notificação de eventos, gerenciamento de metadados, versionamento;
- Alto nível: *workflow*, gerenciamento do banco de dados, personalização;
- Aplicação: alinhamento de uma seqüência genética, um algoritmo de busca genética, o banco de dados Swiss-Prot.

Cada camada está baseada em metadados. Para atingir uma flexibilidade, os serviços da Grid requerem informações sobre a funcionalidade, disponibilidade e interfaces dos vários serviços. Metadados são a chave para alcançar essa visão dos serviços da Grid.

O desenvolvimento da Grid foi focado em itens básicos de gerenciamento do armazenamento, computação e recursos necessários para fazer as informações e ferramentas

de uma comunidade científica global acessíveis em um ambiente de alto desempenho. Entretanto, de um ponto de vista da ciência na Web, o propósito da Grid é entregar um ambiente de colaboração e suporte que permita cientistas geograficamente distribuídos alcançar objetivos de pesquisa mais eficientemente.

O projeto MyGrid [43] foi desenvolvido para desenhar, desenvolver e demonstrar funcionalidades de alto nível sobre a infra-estrutura existente da Grid que dá suporte a cientistas no uso de recursos complexos distribuídos. Esse projeto desenvolve um ambiente que suporta: o processo científico de investigação experimental, acúmulo de evidências e assimilação de resultados; o uso dos cientistas das informações da comunidade e a colaboração científica, permitindo agrupamentos dinâmicos atacar problemas emergentes de pesquisa.

Essa estrutura dá suporte individual aos cientistas provendo facilidades personalizadas relacionadas à seleção de recursos, gerenciamento dos dados e encadeamento dos processos. MyGrid desenvolveu dois ambientes: um que suporta cientistas individuais na análise de dados genéticos funcionais e outro que suporta as anotações de um banco de dados padrão. Ambas tarefas requerem representação explícita e encadeamento dos processos científicos e têm como desafio os requisitos de desempenho.

O foco do projeto MyGrid está nos dados científicos utilizados através da Web e na disponibilização de um ambiente distribuído que apoie o processo experimental *in silico*. A construção de repositórios contendo valores agregados e seus usos em pesquisas do dia-a-dia só será realmente viável quando cientistas tiverem ferramentas eficientes que os permitam unir as diversas bases de dados e ferramentas analíticas, extrair informações relevantes de textos livres e ter recursos computacionais disponíveis para tarefas que precisam usar intensamente o processador. O projeto MyGrid tem como objetivo construir uma infra-estrutura para cientistas que disponibilize um ambiente personalizado contendo problemas e soluções. A visão é de um ambiente como um "livro de laboratório" onde os cientistas possam construir, buscar e adaptar experiências *in silico*, armazenar parcialmente resultados em repositórios locais de dados e ter suas próprias visões dos repositórios públicos. Além disso, os cientistas seriam mais bem informados das ferramentas e dados diretamente relevantes a seu espaço experimental, com isso, a Grid torna-se baseada no ambiente do cientista, daí o nome MyGrid.

Os objetivos de projeto estão listados a seguir:

- Uma **plataforma aberta extensível,** para interoperabilidade de dados e ferramentas, construída usando Web Services e tecnologia de Grid, explorando as habilidades coletivas e as tecnologias em bases de dados, gerenciamento de informações e conhecimento, tecnologias de interoperação e sistemas de múltiplos agentes;
- Suporte para experiências in silico baseado no **fluxo dos processos**, construído utilizando técnicas já disponíveis na comunidade de Web Services;
- Suporte para **disponibilização de dados** e **gerência de mudança de recursos** baseado em notificações e evolução do fluxo dos processos;
- Suporte para **personalização** baseado no gerenciamento de visões dos repositórios de informação e personalização do fluxo dos processos;
- Um **conjunto de ferramentas** que possa ser configurado para aplicações específicas;
- Uma **demonstração de uso**, incorporando recursos de informação apropriados e ferramentas de análise ao MyGrid, capaz de povoar modelos com processos experimentais, dados e resultados pessoais.

O MyGrid tem um modelo em três camadas que consiste em: uma grade computacional de dados, uma grade de serviços de informação e uma grade de informações científicas via Web. As primeiras duas camadas são genéricas (isto é, não há foco específico na representação de artefatos científicos e com isso eles podem ser usados igualmente em aplicações diferentes); a terceira camada proporciona suporte específico para atividades dos cientistas via Web. Essa abordagem envolve uma estratégia incremental: primeiro são tratados problemas da forma mais simples possível e posteriormente esses problemas são incrementados para formarem funcionalidades mais complexas.

Esse projeto está preocupado com o problema de compartilhamento coordenado de fontes de dados e soluções na escala global para aplicações com intensa quantidade de dados e computação. Nesse ambiente o compartilhamento do poder computacional é o principal objetivo. Para isso será necessário que os cientistas realizem algum tipo de mapeamento. A utilização da ferramenta de suporte ao mapeamento que desenvolvemos se adequa a esse ambiente, uma vez que utiliza o formato XML e seus documentos de gerência e armazenamento de informações.

3.3. Pesquisas de Bioinformática no LLNL

Os principais desafios relacionados às pesquisas de bioinformática no Lawrence Livermore National Laboratory (LLNL) [44] na última década são:

- Interagir com um número crescente de usuários;
- Convencer a comunidade científica a investir em novas tecnologias;
- Possibilitar adaptação às mudanças das técnicas de laboratório;
- Integrar novas tecnologias de computação;
- Refletir novas visões dos dados;
- Desenvolver novas técnicas de aquisição de dados;
- Melhorar a confiabilidade dos dados.

Os itens mais complexos e demorados de resolver desses desafios são as dificuldades na utilização de dados dinâmicos e heterogêneos; o desenvolvimento de sistemas práticos para prover acesso efetivo a dados externos e permitir interações mais ricas com os dados.

Com o objetivo de resolver os três itens mencionados anteriormente foi criado o projeto DataFoundry [49]. O alvo desse projeto é melhorar a interação dos cientistas com seus dados. A tarefa inicial foi desenvolver uma infra-estrutura que permitiria o time de bioinformática criar e manter uma visão consistente das diversas fontes de dados autônomas.

Para conseguir isso, DataFoundry desenvolveu uma infra-estrutura baseada em metadados para suportar uma arquitetura de *data warehouse* (DW) com mediadores (figura 3-3). As fontes de dados contêm os dados que devem ser integrados no DW através dos *wrappers* (que validam os dados) e dos mediadores (que traduzem os dados para a representação do DW). O DW é um grande repositório de dados, normalmente um banco de dados relacional, que apresenta uma visão consistente dos dados disponíveis das fontes. Os usuários interagem com o DW através de um conjunto de interfaces customizadas.

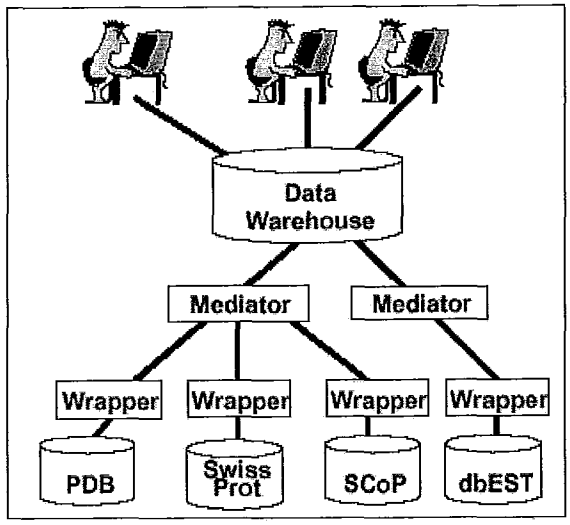


Figura 3-3: Arquitetura Data Warehouse [44]

O desafio de criar um DW para o ambiente genético está em desenvolver uma infra-estrutura suficientemente flexível para lidar com a natureza dinâmica do domínio. Diferentemente das aplicações comerciais, fontes de dados científicas são extremamente dinâmicas. Sempre que uma fonte muda seu formato de dados, o *wrapper* e o mediador devem ser adaptados para lidar com essa nova representação. Para mudanças radicais, ou quando uma nova fonte é integrada, o DW e as interfaces podem também precisar ser atualizadas. Isso torna um extremo desafio manter um DW funcionando quando um grande número de fontes está sendo integrado.

A infra-estrutura do DataFoundry contém um programa gerador de mediadores (MG) que automaticamente gera um mediador usando uma coleção de metadados declarativos. Além disso, o MG define uma classe de biblioteca que pode ser usada pelo *wrapper* para representar os dados obtidos da fonte. Isso simplifica a integração de novas fontes, uma vez que o administrador precisa apenas definir os metadados apropriados e escrever um *wrapper* usando as classes resultantes, em vez de escrever o *wrapper* e o mediador do início. Isso também simplifica a manutenção do DW, uma vez que os metadados são significantemente mais fácies de atualizar que o mediador. O MG também é utilizado para integrar duas novas fontes de dados no DW existente contendo dados de outras duas fontes. Para interagir com o DW, os cientistas usam uma das várias interfaces dependendo da consulta e do nível de *expertise*.

Nesse projeto aparece novamente a necessidade de encadeamento de processos e troca de informações. Para que utilizar a ferramenta de suporte ao mapeamento que desenvolvemos, seria preciso converter as fontes de dados em XML ou em um Web Service. Depois o resultado do mapeamento seria armazenado no DW para futuras conversões.

3.4. Projeto Bio-AXS

O projeto "Bio-AXS" [6, 51, 59] em desenvolvimento pela PUC-Rio, propõe uma arquitetura para integração de fontes de dados e aplicações da biologia molecular. Esse projeto está relacionado ao problema de definição de um modelo de dados para representar e integrar dados genéticos. Um problema bem conhecido dos usuários de dados biológicos, moleculares e genéticos é que a informação que está distribuída em vários *sites* na *Web* se apresenta em diversos formatos, dificultando a visão em um formato único e uniforme. Cada grupo de pesquisa que, no seu dia-a-dia, gera e processa esses dados, normalmente trabalha de uma maneira independente, usado diferentes modelos de dados para representar e manipular basicamente a mesma informação.

A maneira usual para lidar com esse problema de integração de informação é usar um modelo e sistema de integração específico que deve capturar todos os dados necessários para uma aplicação em particular. Como esta é uma área de pesquisa que está em constante mudança, com novas informações estruturais sendo incorporadas e aliadas aos novos requisitos das aplicações, é muito difícil decidir por um único modelo de dados. Além de que, qualquer estratégia baseada em um modelo próprio escolhido, não atende a todas as necessidades de todos os usuários.

Nesse projeto é proposto o uso de um *framework* orientado a objetos para lidar com esse problema de integração de dados genéticos. Um *framework* é um *software* de sistema incompleto que contém componentes básicos pré-definidos (*frozen spots*) e outros que devem ser instanciados (*hot spots*) para a implementação de uma funcionalidade particular desejada. O *framework* proposto pode ser considerado específico para o domínio da aplicação, nesse caso, a área de biologia molecular e genética.

A idéia básica para a escolha do *framework* se deve à necessidade de uma ferramenta para integração de informações genéticas que estão espalhadas em um ambiente distribuído (grande maioria na *Web*). Essas informações estão constantemente mudando e são utilizadas por aplicações distintas. Com isso, certas propriedades, como flexibilidade e ser extensível, aliadas com reutilização são necessárias. O modelo de dados biológicos, uma vez inicialmente definido, precisa agregar informações de diferentes fontes de dados. Através de um *framework*, torna-se possível gerar interfaces e instâncias de bancos de dados, executando as mais importantes aplicações genéticas de um modo uniforme e integrado.

O *framework* proposto propicia [51, 52]:

- Flexibilidade

    - Captura dos esquemas das fontes de dados da biologia
    - Definição e manutenção de um esquema próprio
    - Definição de um modelo de dados / ontologia efetivamente usada nas fontes de dados existentes
    - Utilização das aplicações disponíveis
- Alto desempenho no acesso aos dados
- Extensibilidade
    - Incorporação de qualquer aplicação existente
    - Incorporação de qualquer fonte de dados de biologia
    - Instanciação de uma fonte de dados para uma pesquisa específica
- Tratar a evolução dos esquemas das fontes de dados
    - Detecta alteração de esquemas, via agente de monitoração
    - Informa ao usuário administrador que houve alteração
    - Usuário administrador procede a uma nova captura, no momento adequado. A alteração de esquemas é assíncrona.
- Tratar a evolução dos esquemas específicos
    - A qualquer momento, por ação do administrador
- Tratar a atualização das instâncias de dados
    - Monitora atualização da fonte de dados
    - Procede à alteração de forma autônoma
    - Termina atualização por ação do administrador

O *framework* proposto provê seis funcionalidades básicas:

1. Captura dos esquemas de fontes de dados existentes e diversas;
2. Comparação dos objetos da arquitetura com os objetos no esquema capturado;
3. Captura dos dados pertencentes às diversas fontes de dados;
4. Definição de novos esquemas sob demanda;
5. Geração de dados no formato desejado pela aplicação de biologia molecular;
6. Execução de algoritmos instanciados como métodos de classes do modelo de biologia molecular.

A primeira funcionalidade assume que existe um conversor (*wrapper*) para as fontes de dados sendo consideradas. Quando a segunda funcionalidade não é diretamente obtida, um novo objeto biológico é criado e as associações pertinentes também devem ser criadas. Esse casamento pode estabelecer relacionamentos tais como "*is_synonym_of*". A habilidade de capturar dados é a terceira funcionalidade mencionada e é necessária uma vez que o esquema

associado já foi capturado. Para isso, há um outro tipo de conversor. Listado em quarto lugar há uma demanda comum por novos esquemas para aplicações específicas, junto com novos conjuntos de dados associados. Como existem múltiplos formatos de armazenamento de dados, alguém precisa convertê-los para um formato particular, obrigatório para uma dada aplicação (por exemplo, formato de arquivo FASTA para programas BLAST). Isto é o que a quinta funcionalidade se refere. Com isso, o *framework* possa estar habilitado a executar todos os métodos envolvidos.

O *framework* proposto se divide em quatro módulos:

- Administrador (*Administrator*)
- Capturador (*Captor*)
- *Driver* de aplicação (*Application Driver*)
- Conversor (*Converter*).

A figura 3-4 mostra uma visão geral da arquitetura do *framework.*

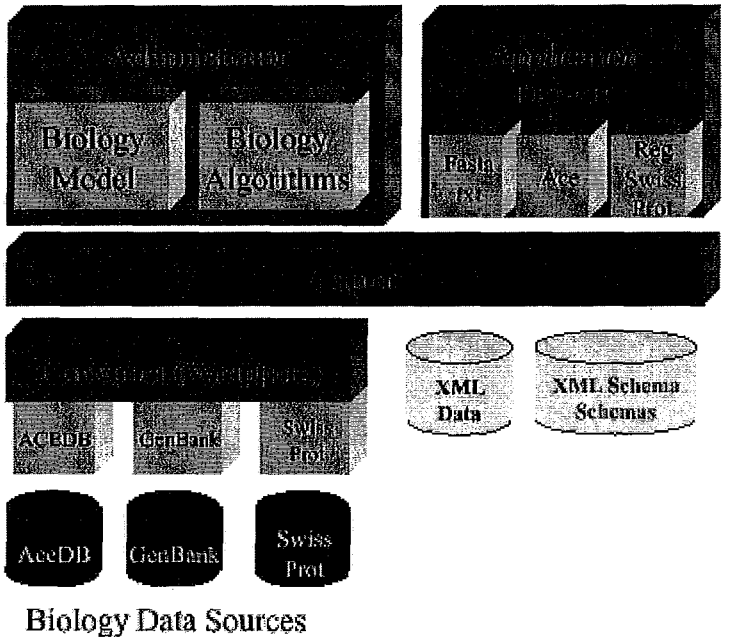


Figura 3-4: Arquitetura do framework Bio-AXS [6]

Os modelos biológicos (*Biology Model*) e algoritmos (*Biology Algorithms*), conversores (*Wrappers*) associados a fontes de dados biológicos e *drivers* das aplicações (*Application Drivers*) são os "*hot spots*" do *framework.* Quando instanciados, eles implementam uma funcionalidade particular, definindo uma aplicação sobre o domínio da biologia molecular.

O módulo administrador (*Administrator*) faz a interface com os usuários de modo a prover gerenciamento dos modelos de dados biológicos, pedidos para capturar esquemas e/ou dados e permite a execução de algoritmos instanciados no próprio *framework.* Além disso,

esse módulo contém um modelo de classes biológicas que é combinado com as fontes de dados existentes, assim como com os métodos que estão associados a essas classes. O módulo capturador (*Captor*) é responsável pelo repositório de dados e arquitetura de esquemas. O módulo conversor (*Converter*) provê acesso às fontes de dados biológicas, fazendo a tradução dos esquemas fontes para a linguagem padrão XML Schema [9] e dos dados para XML [8]. Finalmente, o módulo dos *Drivers* implementa a geração da interface entre as aplicações biológicas e o *framework*.

Por exemplo, se um usuário solicitar um pedido ao módulo administrador para capturar esquemas e/ou dados de uma fonte de dados dada, o administrador manda o pedido para o capturador, que por sua vez o envia para o conversor da fonte de dados biológica sendo considerada. A captura dos esquemas ou dados pode ser finalizada somente se o conversor correspondente tiver sido previamente desenvolvido e instanciado no *framework*. O conversor implementa o mapeamento dos esquemas das fontes de dados para XML Schema e dos dados para XML. Ambos os esquemas e os dados capturados são armazenados em seus respectivos repositórios.

O usuário pode também pedir ao módulo administrador a geração de um arquivo para uma aplicação biológica específica. Muito parecido com o que foi descrito anteriormente, tal pedido somente pode ser pedido se o *driver* associado tiver sido previamente desenvolvido e instanciado na arquitetura. O módulo administrador dispara o *driver* de maneira a executar a tarefa solicitada. Então, o *driver* pede os dados ao capturador que manipula o repositório de dados. Por exemplo, há várias aplicações biológicas que têm como entrada o nome de um arquivo e sua localização para proceder com uma execução.

A arquitetura também permite, via módulo administrador, a execução do algoritmo biológico instanciado na arquitetura, o qual pode trabalhar com os dados armazenados no repositório. A construção de interfaces entre o *framework* e as aplicações biológicas existentes pode ser feita também pelo *driver* da aplicação. Outros serviços podem ser desenvolvidos externamente.

O módulo administrador do *framework* faz a interface com os usuários gerenciando os modelos de dados biológicos, capturando esquemas e/ou dados e permitindo a execução de algoritmos instanciados no próprio *framework*. Para realizar algumas dessas tarefas, os usuários precisam informar ao *framework* o mapeamento de esquemas de dados e de entradas e saídas de programas para permitir a execução dos algoritmos. Para dar ao usuário o suporte necessário nesse processo de mapeamento, a arquitetura Bio-AXS se beneficiaria com a utilização da ferramenta de suporte ao mapeamento de esquemas genéticos que

desenvolvemos, já que o usuário teria auxílio para definir seu mapeamento. Uma vez definido o mapeamento, a ferramenta faria a conversão automaticamente. De fato, o Bio-AXS foi um dos motivadores dessa dissertação de mestrado.

Nossa ferramenta seria um serviço externo da arquitetura Bio-AXS que pode ser chamada pelo módulo administrador do *framework*. Quando o usuário terminasse o processo de mapeamento, nossa ferramenta retornaria as informações necessárias para o módulo administrador do *framework*. Essas informações do mapeamento realizado seriam armazenadas no *framework* e poderiam ser utilizadas por ele sempre que fosse necessário.

3.5. SRMW - Gerenciamento de Recursos Científicos baseado em Web Services

A cooperação entre os cientistas ocorre através da troca de recursos científicos, tais como: dados, programas e modelos matemáticos. Isso é particularmente verdade para aplicações ambientais. Encontrar o recurso certo para aplicar em um problema ambiental é uma tarefa difícil. Normalmente, essa decisão é baseada em experiências passadas. Para facilitar essa troca, reuso e disseminação de informação Cavalcanti [37, 45, 53, 62] propõe uma arquitetura SRMW para gerenciamento de recursos científicos distribuídos.

Para que os recursos científicos distribuídos se tornem parte do grande sistema de informação na Internet, eles precisam ser localizados, entendidos e eficientemente acessados pela rede. Compartilhamento de dados científicos requer identificação dos dados e dos modelos e programas (implementações de modelos) que podem ser usados, quando (e em que ordem) devem ser executados. Além disso, também requer acesso remoto aos dados e gerenciamento da carga para execução na localização do programa. Um passo nessa direção, inclui prover descrições de modelos e programas para facilitar a seleção do modelo apropriado e consequentemente do programa apropriado. Entretanto, isso não é uma tarefa simples. Antes de tudo, desenvolvedores de modelos vêm de diferentes áreas, lidam com diferentes tipos de padrões para modelos e descrições. Além disso, cada descrição de modelo deve incluir suas próprias condições de uso, isto é, regras contextuais e operacionais, as quais são complexas de formalizar. Finalmente torna-se importante o registro dos experimentos que são realizados para facilitar a compreensão e reutilização de modelos escolhidos.

O conceito principal atrás da arquitetura de Web Services (figura 3-5) é a noção de publicação de serviços pela Web para serem usados por outros programas. Mais do que lidar com uma comunicação entre componentes de um programa, a arquitetura de Web Services também se destina à descrição do programa (WSDL – Web Services Description Language) e à busca do programa (Web Services Registry). Entretanto, apenas Web Services não são suficientes para fazer estes recursos realmente úteis. Devem ser incluídos mecanismos para a descrição e a gerência de tais recursos através de uma arquitetura que combine metadados com estes recursos. Em SRMW, este acoplamento é feito associando-se categorias de programas ao código legado disponível e associando-se categorias de dados aos dados disponíveis.

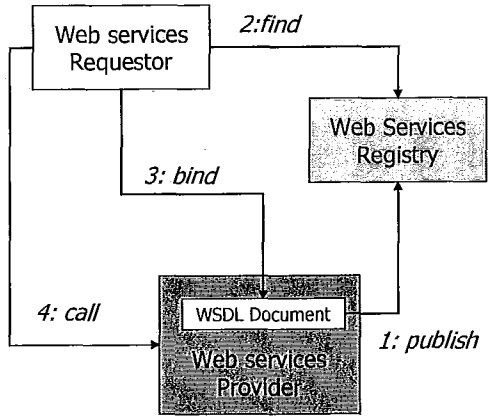


Figura 3-5: Arquitetura Web Services

Na arquitetura SRMW (figura 3-5), o usuário alvo é o usuário científico da aplicação, que deve ser capaz de descrever os dados e programas em aplicações baseadas em workflow. Para fornecer tal funcionalidade, SRMW adiciona alguns módulos à arquitetura de Web Services. Os módulos principais da SRMW são "Web Services Provider" e "Web Services Registry". O "Web Services Provider" é usado para fornecer o acesso aos recursos operacionais, tais como conjuntos de dados e códigos executáveis, pela Web. No caso dos códigos legados, para cada código deve haver um adaptador de Web Service (code wrapper) que recebe pedidos de invocação de acordo com um protocolo uniforme de comunicação (por ex. SOAP). Da mesma forma, para cada conjunto de dados deve haver um adaptador de Web Service (data wrapper) que recebe pedidos dos dados de acordo com o mesmo protocolo.

O "Web Services Registry" corresponde a um catálogo de Web Services que registra descrições de acordo com o metamodelo SPMW (Scientific Publishing Metamodel for Web Services) [45]. Esse metamodelo foi projetado para acomodar metadados sobre os recursos científicos, descrevendo um programa como uma transformação dos dados que requer uma entrada e produz uma saída, associando esse programa aos modelos científicos como sua implementação; descrevendo uma entrada/saída do programa como categorias de dados; descrevendo um code wrapper como um recurso operacional associado a uma descrição do programa; e descrevendo um conjunto de dados como um recurso operacional associado a uma categoria de dados. O "Web Services Registry" permite também descrições de alto nível, tais como as especificações de workflow científicos, que são descritas nos termos de descrições do programa, e não estão sujeitos aos recursos de código disponíveis.

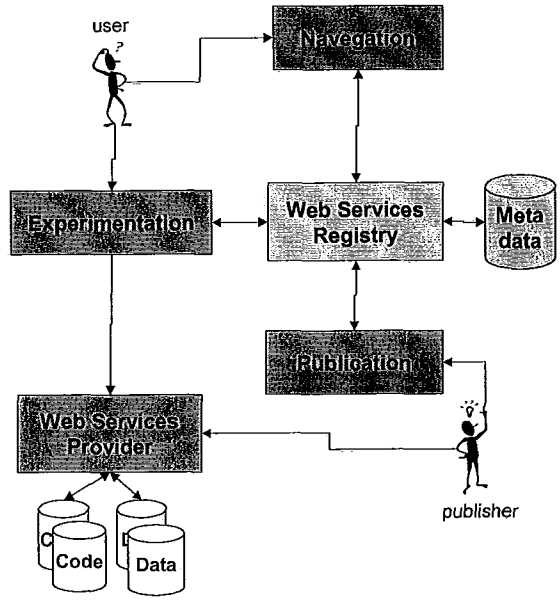


Figura 3-6: Gerenciamento de recursos científicos baseado em Web Services [45]

Os três outros módulos, publicação, experimentação e navegação, fornecem as interfaces aos usuários dos módulos anteriores. O módulo da publicação é usado para incrementar o "Web Services Registry" com metadados científicos. O módulo da navegação é usado para navegação em todos os metadados científicos disponíveis. Finalmente, o módulo de experimentação é usado para executar e registrar as experiências baseadas nestes metadados. Estas experiências são executadas com as instanciações do workflow, que envolvem a associação de cada programa na especificação do workflow a uma localização específica do Web Service (code wrapper). Então, o módulo da experimentação controla os pedidos, trabalhando como um Web Service Requester que constrói pedidos para pegar dados de entrada e executar um código específico com esses dados. Os resultados dos dados estão (uma escolha do usuário de temporariamente ou definitivamente) disponíveis também como um Web Service.

Nossa ferramenta de mapeamento de esquemas genéticos é fundamental para que essa arquitetura possa armazenar e trocar dados entre os programas que compõem o workflow sendo gerenciado pelo SRMW. Realizamos um estudo de caso para validação do nosso trabalho utilizando essa arquitetura. Maiores detalhes serão explicados no próximo capítulo.

CAPÍTULO 4. A FERRAMENTA: DESCRIÇÃO, IMPLEMENTAÇÃO E VALIDAÇÃO

Dentro da COPPE/UFRJ estão sendo desenvolvidos vários trabalhos focados em problemas de gerenciamento de recursos científicos. Atualmente temos três trabalhos que estendem a arquitetura SRMW com três módulos novos [61, 62, 63]: o gerador de Web Services de dados (Data Web Service Generator), o módulo de suporte ao mapeamento entre esquemas de representação de dados genéticos (Mapping Support Tool), que é o resultado dessa dissertação, e o agente do armazenamento de dados (Data Storage Agent) representados na figura 4-1. Estas extensões visam dar um ambiente integrado com a SRMW que permita aos usuários tornar seus dados disponíveis como serviços Web, definir e realizar os mapeamentos necessários e fornecer o armazenamento e a recuperação de resultados intermediários das execuções da experiência, assim permitindo re-execuções parciais de uma experiência.

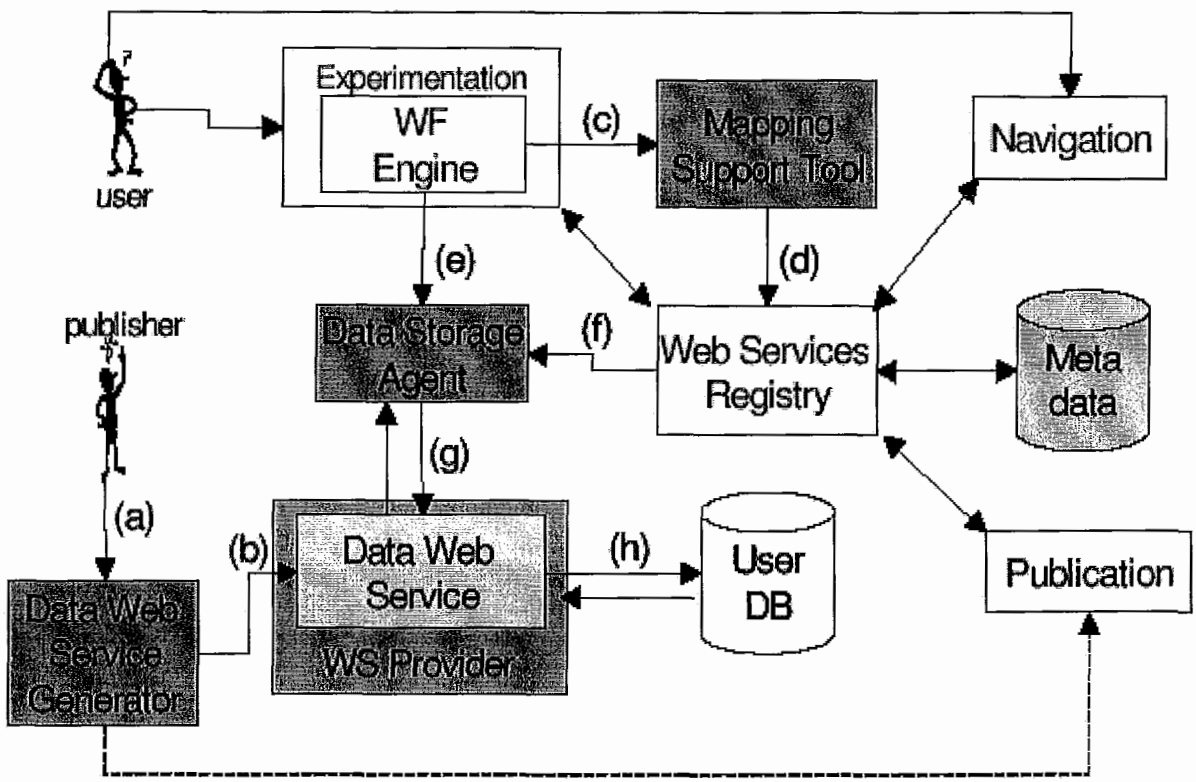


Figura 4-1: Extensões da arquitetura SRMW [63]

Gerador de web services de dados (Data Web Service Generator): Visa permitir que os resultados intermediários do programa (gerados durante a execução de uma experiência) sejam armazenados em um repositório de dados e que estejam disponíveis mais tarde para a recuperação de dados através de um web service na arquitetura de SRMW. Para

isso, é necessário ter um serviço Web de dados que aja como uma interface de acesso a estes dados.

Este serviço Web de dados permite operações de armazenamento e recuperação dos dados. Primeiramente, este módulo cria um serviço Web de dados dentro do provedor de serviços Web. Este serviço Web de dados deve ser registado no registro dos serviços Web da arquitetura SRMW. O módulo da experimentação pode então alcançar esses web services para armazenar resultados dos dados do workflow, ou recuperar seus dados em re-execuções parciais do workflow futuro, através do agente do armazenamento de dados (Data Storage Agent).

O próprio usuário pode construir seu próprio web service de dados baseado na estrutura de sua base de dados. Entretanto, como esta tarefa requer o conhecimento técnico de criar aplicações de banco de dados e serviços Web, está sendo desenvolvido na COPPE/UFRJ um módulo responsável para ajudar o usuário na criação deste serviço Web de dados. Com este módulo, o serviço Web de dados pode ser gerado automaticamente.

Para gerar corretamente o serviço Web de dados, o publicador tem que enviar o esquema da base de dados do usuário ao gerador de serviço Web de dados (figura 4-1-a). Este esquema deve ser descrito como um documento XML com um XML Schema (XSD) associado.

Então, o gerador de serviço Web de dados cria um conjunto das classes de Java que vão compor o wrapper dos dados (figura 4-1-b). Como um serviço Web, o wrapper englobará operações de acesso à base de dados, que serão invocadas pelo agente do armazenamento de dados (figura 4-1-g), de acordo com o pedido de usuário. Este wrapper deve então ser publicado dentro do provedor de serviço Web.

Módulo de suporte ao mapeamento entre esquemas de representação de dados genéticos (Mapping Support Tool): esse módulo é responsável por ajudar o usuário durante o processo de mapeamento entre as diferentes representações dos dados genéticos e gerar automaticamente um arquivo XSL que efetive a conversão. (figura 4-1-d). Este documento será usado pelo agente do armazenamento de dados para converter instâncias de dados em um esquema para instâncias de dados em outro esquema.

Agente do armazenamento de dados (Data Storage Agent): Durante a execução do workflow, para cada dado de saída do programa que o usuário quer armazenar, o agente do armazenamento de dados recebe uma mensagem SOAP de resposta do workflow (figura 4-1-e).

Simultaneamente, o agente do armazenamento de dados, através do registro dos serviços, invoca o repositório de metadados para obter o documento XSL correto gerado pelo módulo de suporte ao mapeamento (figura 4-1-f).

Baseado na mensagem SOAP de resposta e no documento XSL, uma nova mensagem SOAP (SOAP request) é gerada e emitida então ao serviço Web de dados (figura 4-1-g) que armazenará os dados na base de dados do usuário (figura 4-1-h).

4.1. A Descrição da Ferramenta

Nossa proposta de trabalho [61] se encaixa dentro da linha de pesquisa de ambientes integrados para trabalhar com as diversas fontes de dados em seus formatos originais e seus workflows. Apresentamos no capítulo dois duas arquiteturas que se propõem a disponibilizar para a comunidade científica um ambiente integrado para trabalhar com os vários programas e fontes de dados nos seus diversos formatos, propiciando a integração desejada. Essas arquiteturas são: "Projeto Bio-AXS" [6, 51] e "Gerenciamento de Recursos Científicos baseado em Web Services (SRMW)" [37, 45, 53]. Essas arquiteturas motivaram nosso trabalho e o desenvolvimento da ferramenta de apoio ao mapeamento de esquemas de dados genéticos.

A ferramenta de apoio ao mapeamento de esquemas genéticos foi desenvolvida para ser utilizada dentro de uma arquitetura de integração como as mencionadas anteriormente.

Entretanto, projetamos a ferramenta para ser um módulo externo e independente da arquitetura para que possa ser utilizado em vários ambientes. O objetivo é disponibilizar mais um serviço a essas arquiteturas, de modo a ajudar os usuários a fornecer parte das informações requisitadas para o funcionamento desses ambientes.

4.1.1. A ferramenta dentro do Projeto Bio-AXS

Uma das utilizações possíveis do módulo proposto é dentro do *framework* construído no "Projeto Bio-AXS" [6,51]. Algumas das funcionalidades básicas desse *framework* são a captura dos esquemas de diversas fontes de dados e a comparação dos objetos da arquitetura com os objetos no esquema capturado. A primeira assume que existe um conversor (wrapper) para as fontes de dados sendo consideradas. Quando a comparação entre os objetos não é diretamente obtida, um novo objeto biológico é criado e as associações pertinentes também devem ser criadas. Esse casamento pode estabelecer relacionamentos tais como "*is_synonym_of*".

A ferramenta proposta ajudaria na segunda funcionalidade mencionada: a comparação dos objetos da arquitetura com os objetos no esquema capturado. Nossa ferramenta ajudaria dando suporte ao usuário na atividade de definir as associações pertinentes e a estabelecer os relacionamentos de equivalência entre os esquemas.

A utilização da nossa ferramenta de apoio ao mapeamento genético dentro do *framework* proposto pelo projeto Bio-AXS é conseguida acoplando-se nossa ferramenta ao módulo capturador do *framework* que é responsável pelo repositório de dados e arquitetura de esquemas. O módulo administrador do *framework* é o responsável pela interface com os usuários de modo a prover gerenciamento dos modelos de dados biológicos, pedidos para capturar esquemas e/ou dados e a execução de algoritmos instanciados no próprio *framework.*

Quando o usuário desejar integrar diferentes modelos, o módulo administrador manda uma solicitação para o módulo capturador que chama o módulo de suporte ao mapeamento de esquemas. Através da nossa ferramenta, agora como um módulo do *framework*, o usuário pode escolher os modelos desejados, visualizá-los em um formato de fácil entendimento e integrá-los interativamente. Nosso módulo devolve ao capturador as informações de equivalência entre os esquemas de acordo com as associações feitas pelo usuário.

A utilização da ferramenta dentro da arquitetura Bio-AXS descrita anteriormente foi projetada, mas ainda não foi efetivamente desenvolvida.

4.1.2. A ferramenta dentro da Arquitetura SRMW

Outro ambiente que motivou nosso trabalho foi a arquitetura de gerenciamento de recursos científicos baseado em Web Services: SRMW [37, 45, 53], apresentada anteriormente no capítulo dois.

Qualquer pessoa pode disponibilizar seus dados e programas genéticos, publicando-os em Web Services espalhados pela Web. A arquitetura SRMW é importante quando o usuário desejar integrar diferentes serviços Web, ou seja, utilizar seus dados ou programas com outros dados ou programas publicados em outro serviço Web também disponível na Web.

O módulo de experimentação dessa arquitetura executa os experimentos baseados nos metadados informados. Os experimentos são executados com instanciações de workflows, que envolvem a associação de cada programa a uma localização específica do serviço Web (*code wrapper*). Então, o módulo da experimentação controla os pedidos, trabalhando como um '*Web Service Requester*' que constrói pedidos para pegar dados de entrada e executar um código específico com esses dados. De fato, o módulo da experimentação seria beneficiado pela utilização do nosso módulo de suporte ao mapeamento de esquemas, a fim fornecer o

mapeamento dos dados genéticos. Ao interagir com o módulo da experimentação, o usuário escolhe um conjunto de dados para ser usado como entrada de um programa. Baseado nesta informação, este módulo verifica com a base de dados dos metadados para ver se as categorias de dados, relacionadas à entrada do programa e ao recurso dos dados, são as mesmas ou se já foram mapeadas entre si. Senão, o módulo da experimentação ativa a ferramenta de suporte ao mapeamento para fornecer o mapeamento entre aquelas categorias de dados. Através do módulo de suporte ao mapeamento, o usuário pode informar as descrições dos Web Services desejados ou informar diretamente os esquemas, visualizar os formatos de dados ou entradas e saídas de programas através de uma forma de fácil entendimento e integrá-los interativamente.

4.2. Cenários de utilização

Quando analisamos os problemas envolvidos na integração de dados genéticos que requerem um mapeamento das informações, podemos distinguir alguns cenários onde a interferência do usuário é necessária. Nessas situações, o usuário necessita de alguma ferramenta que o ajude durante o processo de mapeamento.

O primeiro cenário é o encadeamento de programas e dados genéticos. Por exemplo, temos um programa que apresenta seus dados de saída em um formato e um outro programa que requer seus dados de entrada em um outro formato diferente do primeiro. Nesse caso, os formatos são de alguma forma equivalentes e é desejado que a execução desses programas seja encadeada. Estamos considerando que o mapeamento dessas informações não é trivial e depende do usuário. O usuário precisa informar as equivalências entre os diferentes formatos. De posse dessas informações, deve ser feita uma conversão dos dados de modo a permitir que a execução encadeada dos programas envolvidos seja possível.

Na figura 4-2 podemos ver um desenho desse cenário que envolve o mapeamento de dois esquemas existentes e que não resulta em alterações nesses esquemas. O resultado esperado é o mapeamento dos dados realizado e um agente transformador (XSL) para efetivar o mapeamento, tornando possível a execução dos programas como desejado.

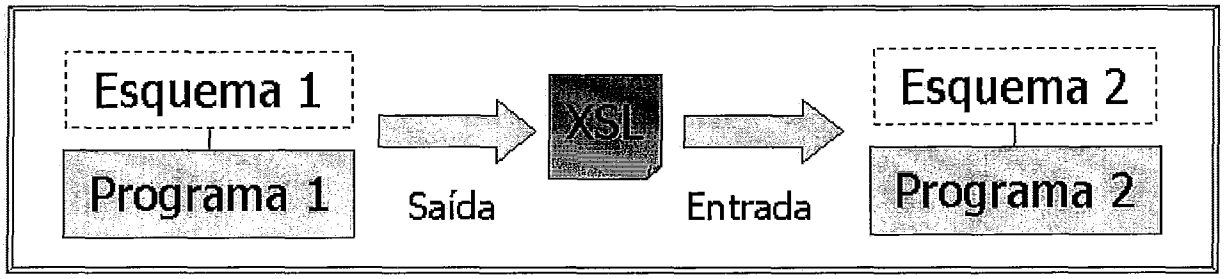


Figura 4-2: Primeiro cenário de utilização

O cenário pode apresentar algumas variantes. Por exemplo, pode existir um conjunto de dados que foram armazenados por um laboratório em um determinado formato. Outro laboratório está interessado em pesquisar esses dados utilizando um determinado programa. A questão é que os dados estão armazenados em um determinado formato e o programa requer os dados de entrada em outro formato diferente do primeiro. É necessário realizar o mapeamento desses formatos e a geração de um agente transformador que efetive a transformação dos dados de modo a tornar possível a execução do programa com os dados desejados sempre que seja necessário esse fluxo.

Existe ainda um problema quando o esquema origem não possui todas as informações necessárias ao esquema destino. Se as informações a mais não são mandatórias, o agente transformador cria os elementos vazios. Mas quando as informações a mais são mandatórias, há necessidade de procurar na arquitetura na qual a ferramenta está acoplada se existe a definição de valores *default* que possam ser utilizados na transformação.

O segundo cenário é a utilização de um repositório global para os diversos esquemas genéticos. Por exemplo, um determinado laboratório de pesquisa deseja reunir os diversos dados utilizados pelos seus vários cientistas em um repositório global. O objetivo é que cada cientista possa enriquecer o repositório incluindo novas informações ou associando as informações que são equivalentes. Para cada novo esquema que se deseja armazenar, deverão ser informadas quais informações devem ser incluídas ou alteradas e as equivalências associadas. Uma vez informado esse mapeamento, o esquema global alterado é salvo no repositório para futuras utilizações. Com essa visão, o repositório global fica cada vez mais enriquecido e serve como um catálogo de todas as informações disponíveis.

Na figura 4-3 podemos ver um desenho desse cenário que envolve o mapeamento de um novo esquema para o repositório global de esquemas, resultando na criação de um novo esquema global ou na alteração do esquema global existente. O resultado esperado é a criação e manutenção de um catálogo com todas as informações desejadas, onde se possa armazenar diversos dados e suas equivalências.

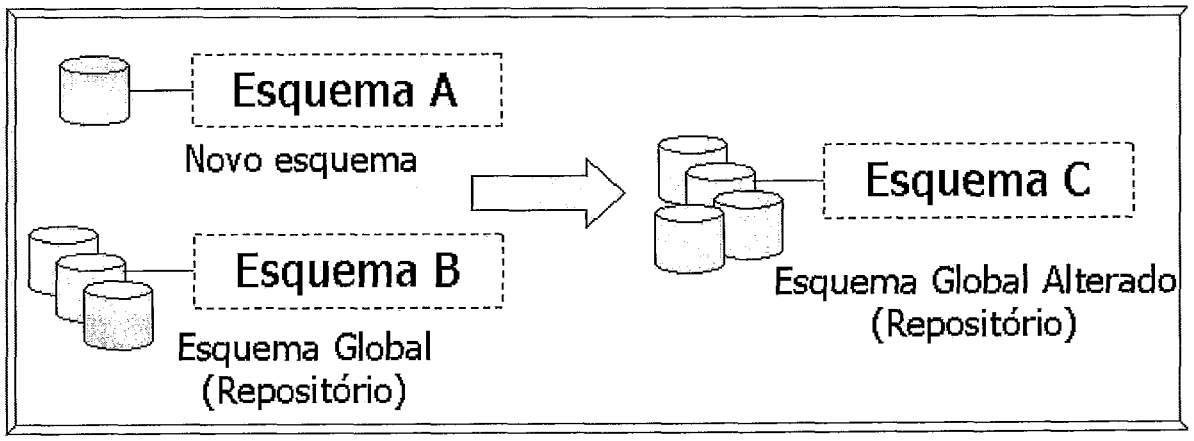


Figura 4-3: Segundo cenário de utilização

Os cenários descritos anteriormente podem apresentar diversas variações de acordo com as necessidades de cada grupo. O importante é notar a necessidade de uma ferramenta para dar suporte à realização da tarefa de mapeamento dessas informações.

4.3. Tecnologias utilizadas

O módulo de suporte ao mapeamento de esquemas genéticos foi desenvolvido utilizando algumas tecnologias que consideramos mais adequadas ao problema tratado:

- Os esquemas genéticos para entrada podem estar em dois formatos: XML Schema e WSDL. XML Schema é praticamente um padrão para dados semi-estruturados devido à sua flexibilidade e possibilidade de integração. As descrições dos Web Services em WSDL também podem ser fornecidas como entrada. O módulo de suporte ao mapeamento extrai os esquemas que descrevem os dados ou programas que estão no formato de XML Schema.
- As informações utilizadas podem ser armazenadas localmente ou no repositório, dependendo da escolha do usuário. Poderíamos adotar vários bancos como repositório, mas optamos por utilizar o banco de dados Oracle 9i. [10], pois possui diversas facilidades para tipos XML, além de disponibilizar algumas bibliotecas para utilização combinada com Java.
- A linguagem utilizada é Java e a conexão com o banco de dados é feita via JDBC [11]. Java é uma linguagem que não depende de um sistema operacional específico. Essa característica é vital para a área genética devido à variedade de sistemas operacionais utilizados pelas ferramentas que estão disponíveis para os pesquisadores.
- O módulo de suporte ao mapeamento gera um arquivo XSL para efetivamente converter os esquemas genéticos servindo como uma ponte para a integração desejada: XSL é um padrão para conversão de documentos XML.

4.4. Ambiente de utilização

Na entrada do módulo, o usuário deverá escolher o ambiente que deseja utilizar o módulo de suporte ao mapeamento. Na figura 4-4 podemos visualizar a tela que será mostrada.

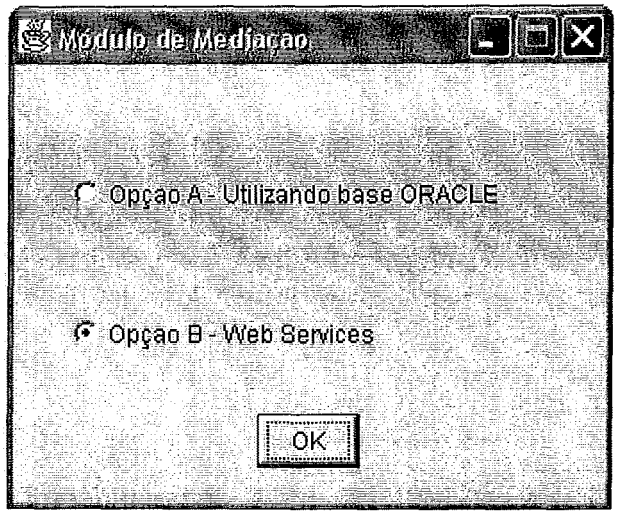


Figura 4-4: Escolha do ambiente

Caso seja escolhida a opção A, o ambiente será o apresentado na primeira proposta do capítulo anterior: utilização com ambiente *Framework* utilizando uma base de dados Oracle para armazenar os esquemas genéticos.

Caso seja escolhida a opção B, o ambiente será o apresentado na segunda proposta do capítulo anterior, para integração de dados e programas publicados em diversos Web Services espalhados pela *Web*.

- Base Oracle

Essa opção foi implementada utilizando um banco Oracle 9i [10]. Essa versão do Oracle possui uma série de facilidades para tratamento de dados XML.

Os documentos XML são armazenados como um objeto XMLDOM. Com isso, não precisamos nos preocupar com detalhes de armazenamento. Se fosse um banco puramente relacional, os documentos XML teriam que ser quebrados em várias tabelas, gerando uma complexidade bem maior no armazenamento e recuperação dos documentos XML. Além disso, existem facilidades de busca dentro dos documentos XML utilizando XPath.

Para exemplificar essse tipo de solução para o repositório, criamos uma base bem simples, com apenas uma tabela chamada ESQUEMA_GENETICO que possui dois campos: NOME_ESQUEMA do tipo VARCHAR2(50) e ESQUEMA do tipo SYS.XMLTYPE.

4.5. Principais classes

No apêndice B são mostrados alguns diagramas da ferramenta desenvolvida. No diagrama de classe podemos ver detalhes das principais classes utilizadas na implementação da ferramenta:

- Principal ⇒ classe que implementa a chamada externa da ferramenta para que seja inicializada.
- TelaEntrada ⇒ classe que implementa a tela que permite a escolha do ambiente de utilização (figura 4-4).
- TelaPrincipal ⇒ classe que implementa a tela que permite a escolha das informações a serem integradas (figura 4-5).
- TreeViewer ⇒ classe que implementa a chamada da classe TreeView.
- TreeView ⇒ classe que implementa a tela principal da ferramenta, onde todas as operações e o mapeamento são realizados (figura 4-7).
- TelaSobre ⇒ classe que implementa a tela com as informação sobre os desenvolvedores da ferramenta.
- Repositorio ⇒ classe que implementa as operações de comunicação com o repositório de esquemas, no caso uma base no banco Oracle 9i.
- MeuAviso ⇒ classe que implementa a tela utilizada para mostrar avisos ao usuário.
- DomTree, DOMParserSaveEncoding, ParseError, Model, FileNameInput, ExampleFileFilter, RadioButtonHandler, ErrorStorer, DefaultImages ⇒ são classes do Apache utilizadas para lidar com objetos XMLDOM e classes auxiliares.

4.6. Escolha de esquemas e Operações básicas

Logo depois da escolha do ambiente na qual deseja utilizar o módulo de suporte ao mapeamento, o usuário deverá escolher os esquemas ou as descrições dos Web Services de origem e destino desejados. Os primeiros deverão estar no formato XML Schema e os segundos no formato WSDL.

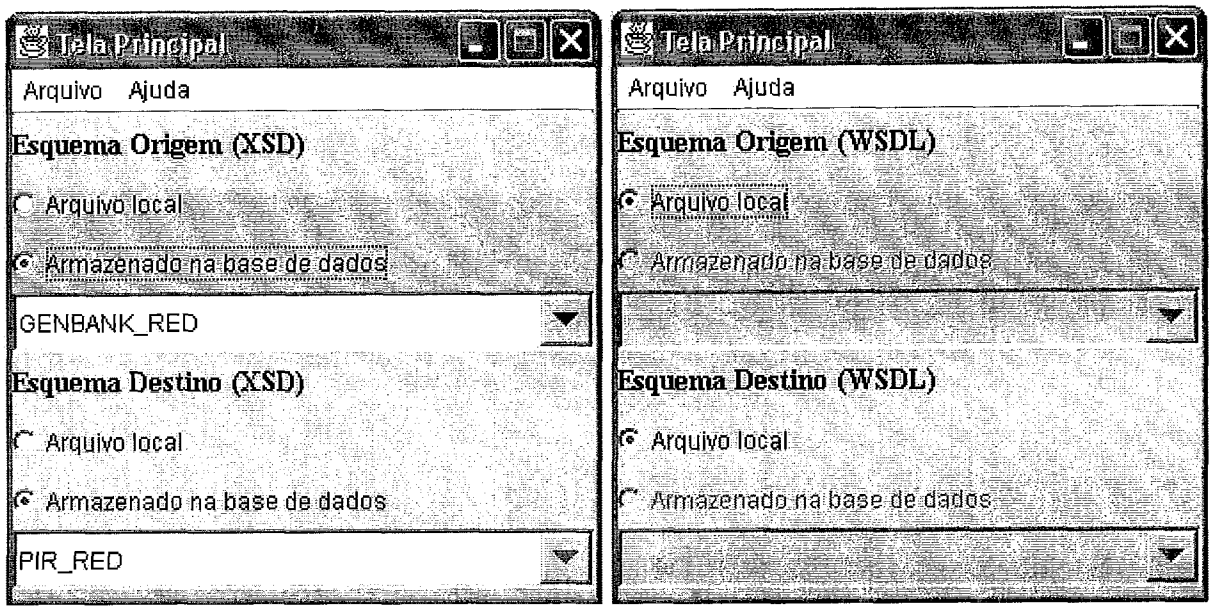


Figura 4-5: Escolha das informações a serem integradas

Podemos ver na figura 4-5, as telas que serão visualizadas pelo usuário caso escolha a opção A ou a opção B, respectivamente.

Caso a opção desejada tenha sido a opção A, os esquemas origem e destino podem ser escolhidos a partir dos esquemas que estão armazenados no repositório ou de novos arquivos no formato XML Schema (XSD) que podem ser selecionados no local escolhido pelo usuário. Caso o esquema ainda não esteja no repositório, será criada uma nova entrada na base de esquemas do repositório, caso contrário o esquema será sobrescrito no repositório.

O fato de trabalhar com esquemas que não estão necessariamente no repositório fornece, além da possibilidade de inclusão de um novo esquema, a facilidade de armazenar localmente um esquema em que se está trabalhando e depois recuperá-lo. Essa facilidade é importante para evitar perda de informação e ganho de tempo para o usuário.

Caso a opção selecionada tenha sido a opção B, as descrições dos Web Services no formato WSDL devem ser selecionadas a partir do local indicado pelo usuário.

Depois das entradas selecionadas, os esquemas genéticos ou descrições dos Web Services que serão mostrados para o usuário são carregados a partir dos arquivos selecionados ou do repositório. Logo depois, as operações podem começar a serem feitas interativamente com o usuário conforme descrito no próximo capítulo. Depois de concluídas as operações, o esquema destino será salvo no repositório ou no respectivo arquivo. Além disso, será gerado um arquivo XSL que serve para a efetiva conversão dos esquemas envolvidos.

Como um XML Schema não deixa de ser um documento XML, podemos encarar que todas as operações envolvem elementos XML, ou seja, vendo de uma maneira simplificada, começam por <tag> e terminam com </tag> ou são vazios <tag/>.

Todas as operações básicas desse módulo de suporte ao mapeamento de esquemas genéticos são bastante visuais, facilitando a sua utilização inclusive para usuários não familiarizados com o padrão XML e XML Schema.

O usuário pode realizar as operações desejadas para agregar as informações do esquema origem no esquema destino, visualizando as alterações feitas no esquema destino passo a passo.

No fundo da tela, o usuário pode ver as informações dos elementos selecionados além de possíveis mensagens de aviso ou de erro.

Além disso, todo o histórico das operações estará sendo guardado em um *log* de operações realizadas conforme podemos ver na figura 4-6:

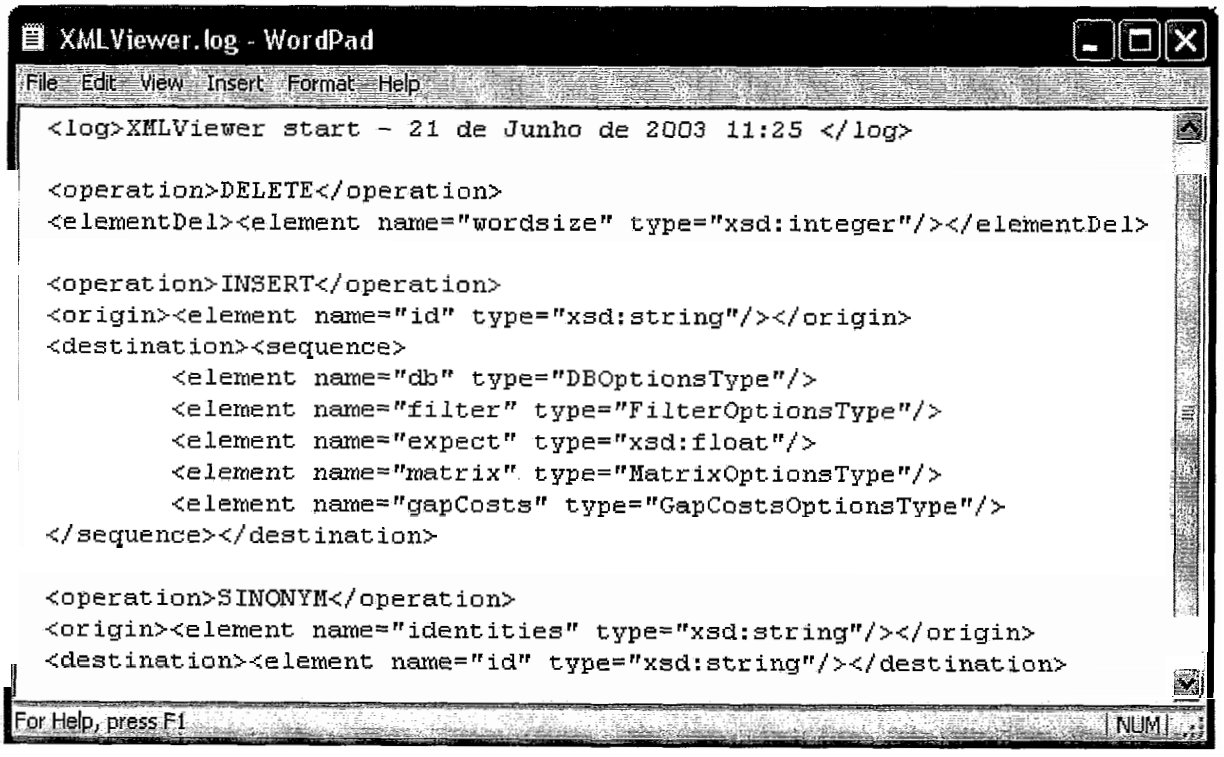


Figura 4-6: Log de operações realizadas

Nas próximas seções vamos explicar detalhadamente as operações disponíveis no módulo em questão.

4.6.1. Inserir um elemento

A idéia dessa operação é inserir um elemento selecionado no esquema origem dentro do esquema destino. Dessa forma podemos agregar definições de novos elementos no esquema desejado.

O usuário deve selecionar o elemento que deseja inserir no esquema destino. Esse é incluído como um sub-elemento do elemento que o usuário selecionar no esquema destino.

Resumindo: os elementos do esquema origem e do esquema destino devem ser selecionados. Todo o elemento origem e seus sub-elementos são incluídos como sub-elementos do elemento destino.

No exemplo a seguir podemos entender melhor a operação explicada anteriormente.

Elemento do esquema origem que se deseja incluir:

```
<xs:element name="accession" type="xs:string"/>
```

Elemento selecionado no esquema destino antes da operação:

```
<xs:element name="base_count">
    <xs:complexType>
        <xs:attribute name="a" type="xs:string" use="required"/>
        <xs:attribute name="c" type="xs:string" use="required"/>
        <xs:attribute name="g" type="xs:string" use="required"/>
        <xs:attribute name="t" type="xs:string" use="required"/>
        <xs:attribute name="others" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

Elemento no esquema destino depois da operação:

```
<xs:element name="base_count">
    <xs:complexType>
        <xs:attribute name="a" type="xs:string" use="required"/>
        <xs:attribute name="c" type="xs:string" use="required"/>
        <xs:attribute name="g" type="xs:string" use="required"/>
        <xs:attribute name="t" type="xs:string" use="required"/>
        <xs:attribute name="others" type="xs:string"/>
        <xs:element name="accession" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

No exemplo anterior, o usuário teria selecionado o elemento accession no esquema origem e o sub-elemento complexType do elemento base_count no esquema destino. Então escolheu a operação "Insere Elemento" na barra de operações. Depois da operação realizada, o elemento accession foi incluído abaixo do sub-elemento complexType do elemento base_count.

Podemos visualizar na figura 4-7 um exemplo da tela para esse tipo de operação.

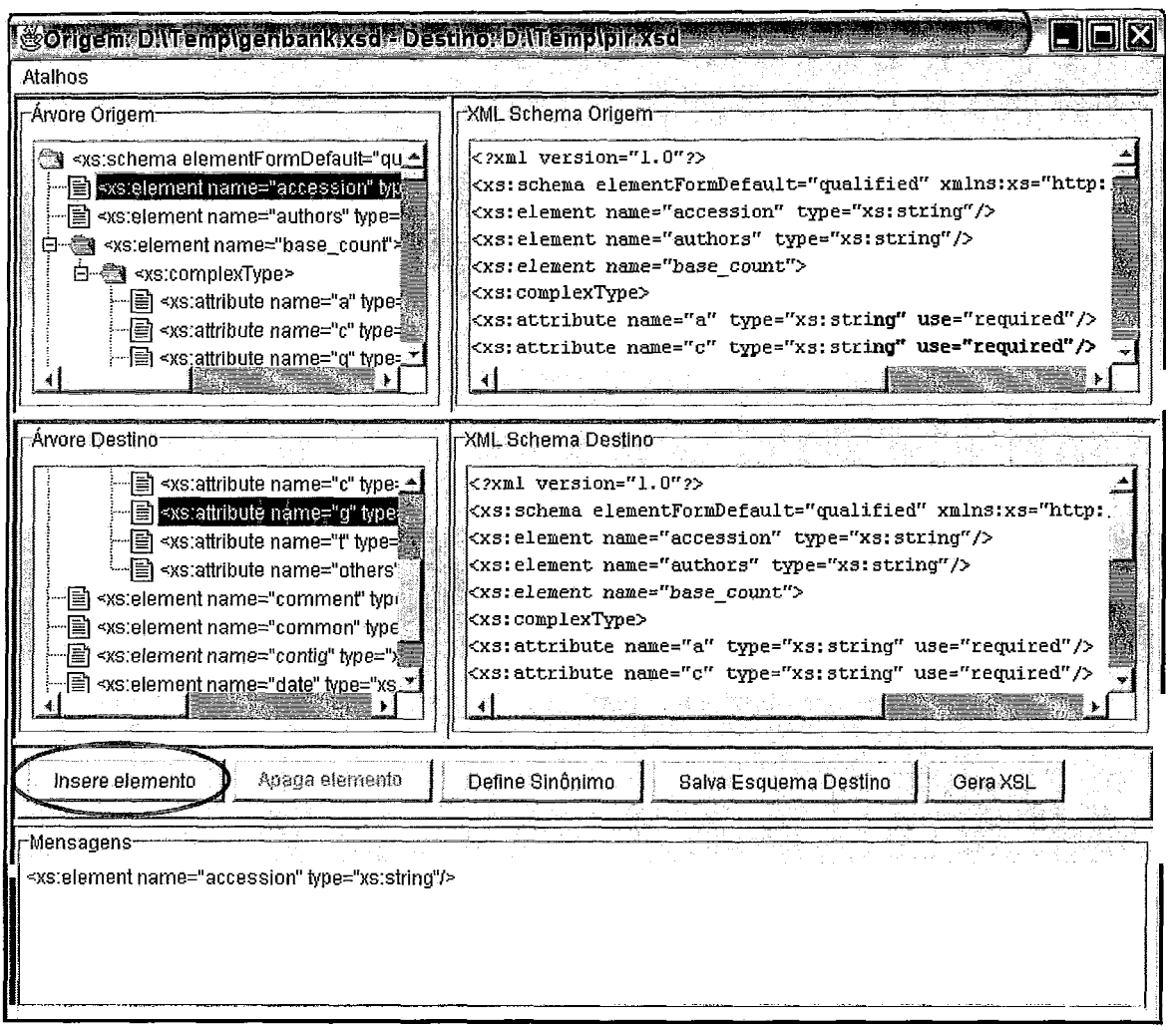


Figura 4-7: Operação "Insere Elemento"

No exemplo mostrado na figura 4-7, vamos inserir o elemento **accession** logo após o elemento **g**.

4.6.2. Excluir um elemento

A idéia dessa operação é excluir um elemento selecionado do esquema destino. Dessa forma podemos retirar definições de elementos indesejáveis no esquema final.

O usuário deve selecionar o elemento que deseja excluir do esquema destino. Resumindo, todo o elemento selecionado e seus sub-elementos são apagados do esquema destino.

No exemplo a seguir podemos entender melhor a operação explicada anteriormente.

Elemento no esquema destino antes da operação:

```
<xs:element name="base_count">
    <xs:complexType>
        <xs:attribute name="a" type="xs:string" use="required"/>
        <xs:attribute name="c" type="xs:string" use="required"/>
        <xs:attribute name="g" type="xs:string" use="required"/>
        <xs:attribute name="t" type="xs:string" use="required"/>
        <xs:attribute name="others" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

Sub-elemento selecionado que se deseja apagar:

```
<xs:attribute name="others" type="xs:string"/>
```

Elemento no esquema destino depois da operação:

```
<xs:element name="base_count">
    <xs:complexType>
        <xs:attribute name="a" type="xs:string" use="required"/>
        <xs:attribute name="c" type="xs:string" use="required"/>
        <xs:attribute name="g" type="xs:string" use="required"/>
        <xs:attribute name="t" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
```

No exemplo anterior, o usuário teria selecionado o sub-elemento others do elemento base_count no esquema destino. Então escolheu a operação "Apaga Elemento" na barra de operações. Depois da operação realizada, o sub-elemento others foi excluído do elemento do elemento base_count.

Podemos visualizar na figura 4-8 um exemplo da tela para esse tipo de operação.

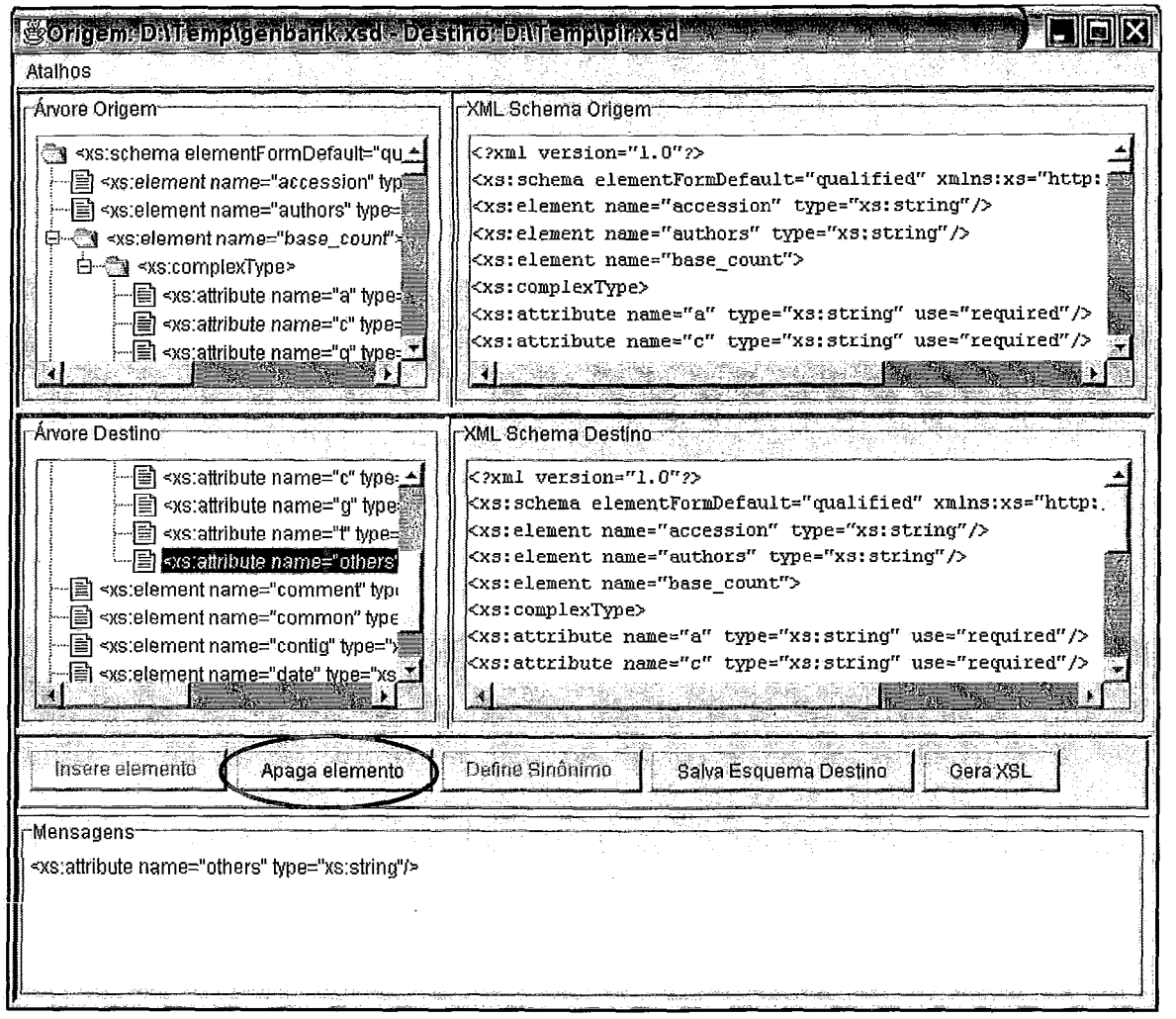


Figura 4-8: Operação "Apaga Elemento"

No exemplo mostrado na figura 4-8, vamos excluir o elemento **others**.

4.6.3. Declarar que um elemento é sinônimo de outro elemento

A idéia dessa operação é documentar que dois elementos diferentes são sinônimos um do outro. Dessa forma podemos informar que duas definições distintas representam o mesmo objeto do mundo real.

O usuário deve selecionar o elemento no esquema origem que deseja documentar como sinônimo do elemento selecionado no esquema destino. Resumindo, será documentado no esquema destino que os dois elementos selecionados são sinônimos. Para isso utilizamos o elemento de documentação do XML Schema.

No exemplo a seguir podemos entender melhor a operação explicada anteriormente.

Elemento selecionado no esquema origem que se deseja declarar como sinônimo:

```
<xs:element name="description" type="xs:string"/>
```

Elemento selecionado no esquema destino antes da operação:

```
<xs:attribute name="comment" type="xs:string"/>
```

Elemento no esquema destino depois da operação:

```
<annotation>
<appinfo>
<synonym>description - comment</synonym>
<source>d:\exemplo.xsd</source>
</appinfo>
</annotation>
<xs:attribute name="comment" type="xs:string"/>
```

No exemplo anterior, o usuário teria selecionado o elemento description no esquema origem e o elemento comment no esquema destino. Então escolheu a operação "Define Sinônimo" na barra de operações. Depois da operação realizada, o elemento de documentação annotation foi incluído abaixo do elemento comment descrevendo que os elementos description e comment são sinônimos.

Podemos visualizar na figura 4-9 exemplos das telas para esse tipo de operação.

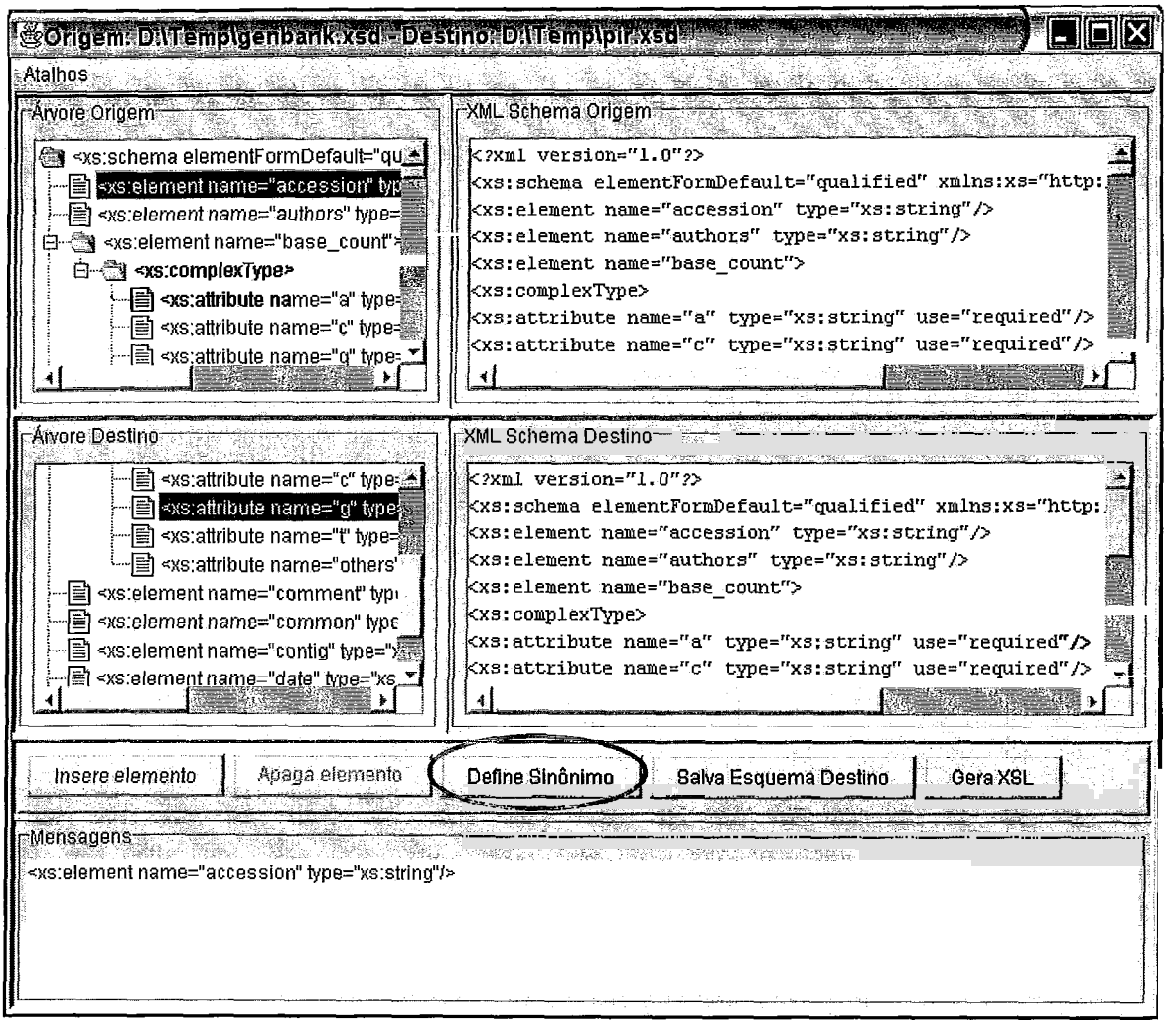


Figura 4-9: Operação "Define Sinônimo" – Opção XSD

No exemplo mostrado na figura 4-9, vamos definir os elementos **accession** e **g** como sinônimos na opção de ambiente que utiliza como entrada arquivos XML Schema (XSD).

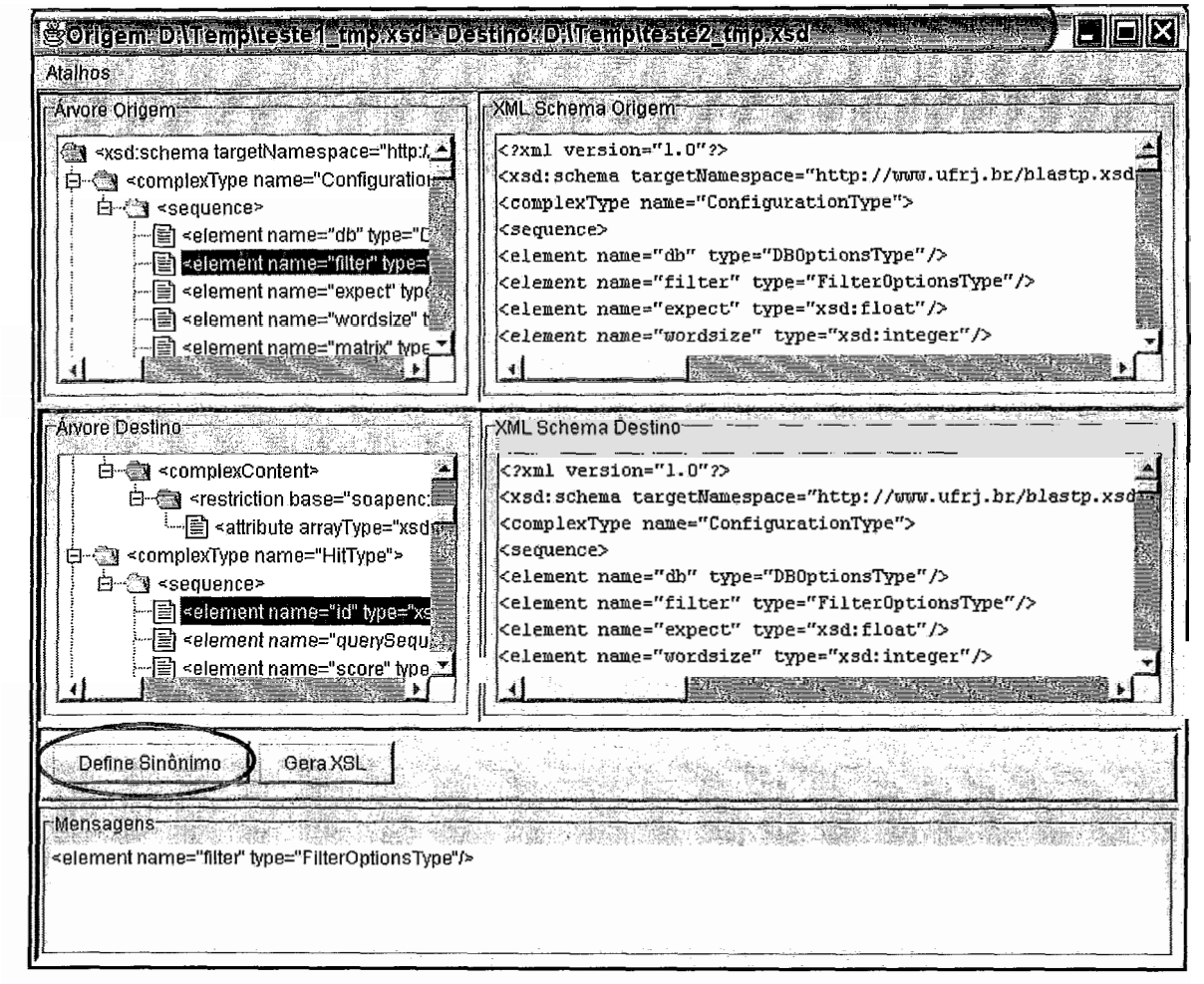


Figura 4-10: Operação "Define Sinônimo" – Opção WSDL

No exemplo mostrado na figura 4-10, vamos definir os elementos **filter** e **id** como sinônimos na opção de ambiente que utiliza como entrada arquivos de descrição de Web Services (WSDL).

4.6.4. Salvar esquema destino

A idéia dessa operação é permitir a qualquer momento salvar o esquema destino, de forma a evitar retrabalho e perda de informação, no caso de acontecer algum imprevisto durante a execução do módulo de suporte ao mapeamento.

Assim que os arquivos de entrada são carregados, estes são copiados para arquivos locais na máquina do usuário. Isso melhora o desempenho do módulo de suporte ao mapeamento e permite que a cada operação, o trabalho seja salvo localmente sem sobrescrever os originais. Além disso, serve como garantia que se algum erro ocorrer, o trabalho não será perdido, visto que podemos fornecer de entrada qualquer arquivo local. Quando o usuário finalizar suas operações, pode salvar o esquema destino na base de dados.

Podemos visualizar na figura 4-11 um exemplo de tela para essa operação.

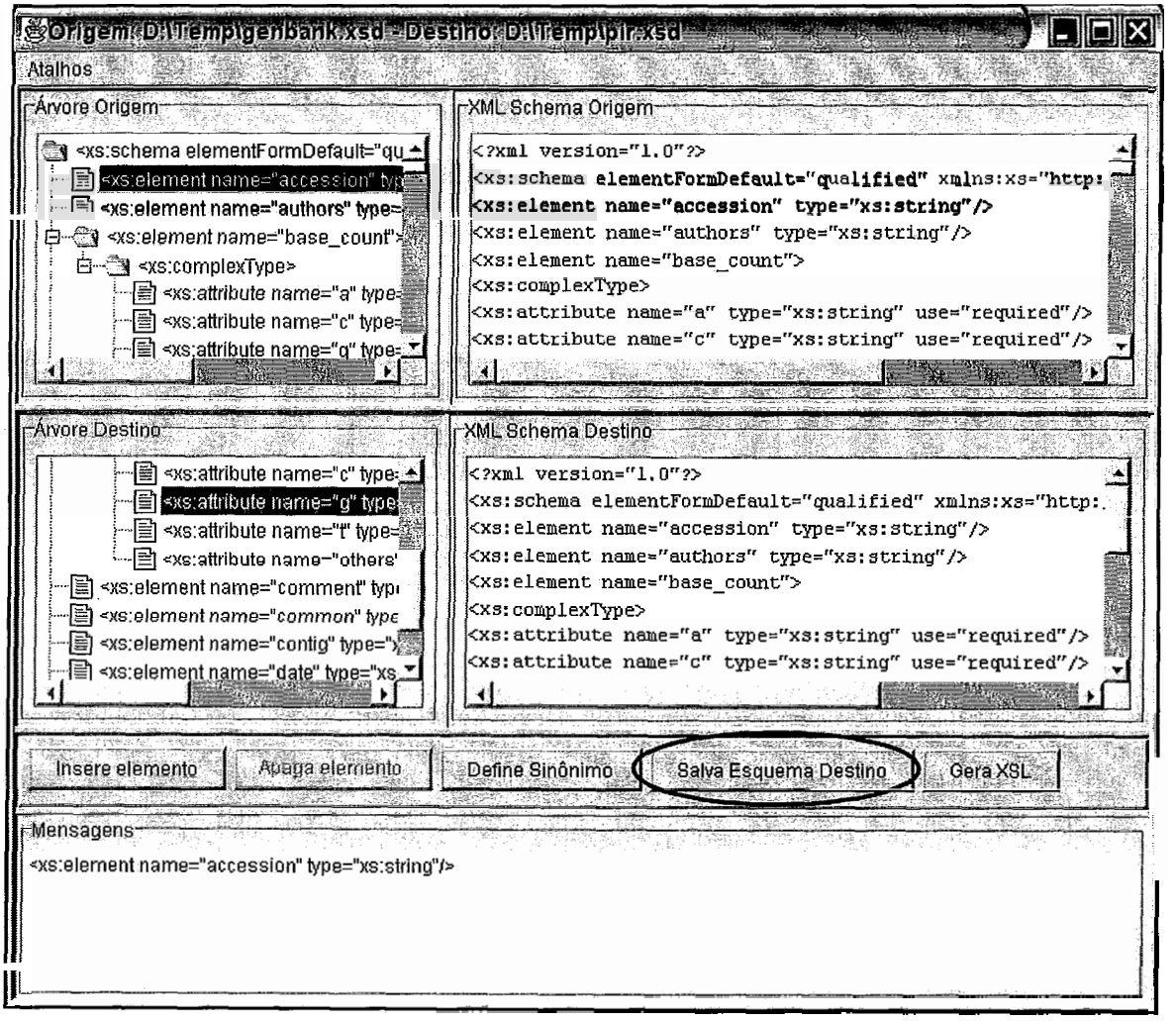


Figura 4-11: Operação "Salva esquema destino"

4.6.5. Geração de arquivo de transformação XSL

A idéia dessa operação é gerar um arquivo de transformação XSL baseado em todos os sinônimos que foram definidos. Com esse arquivo, podemos transformar um documento baseado no esquema origem em um documento no formato do esquema destino. As regras para essa transformação são definidas pelos usuários de acordo com as operações de definição de sinônimo.

Podemos visualizar nas figuras 4-12 e 4-13 exemplos das telas para esse tipo de operação.

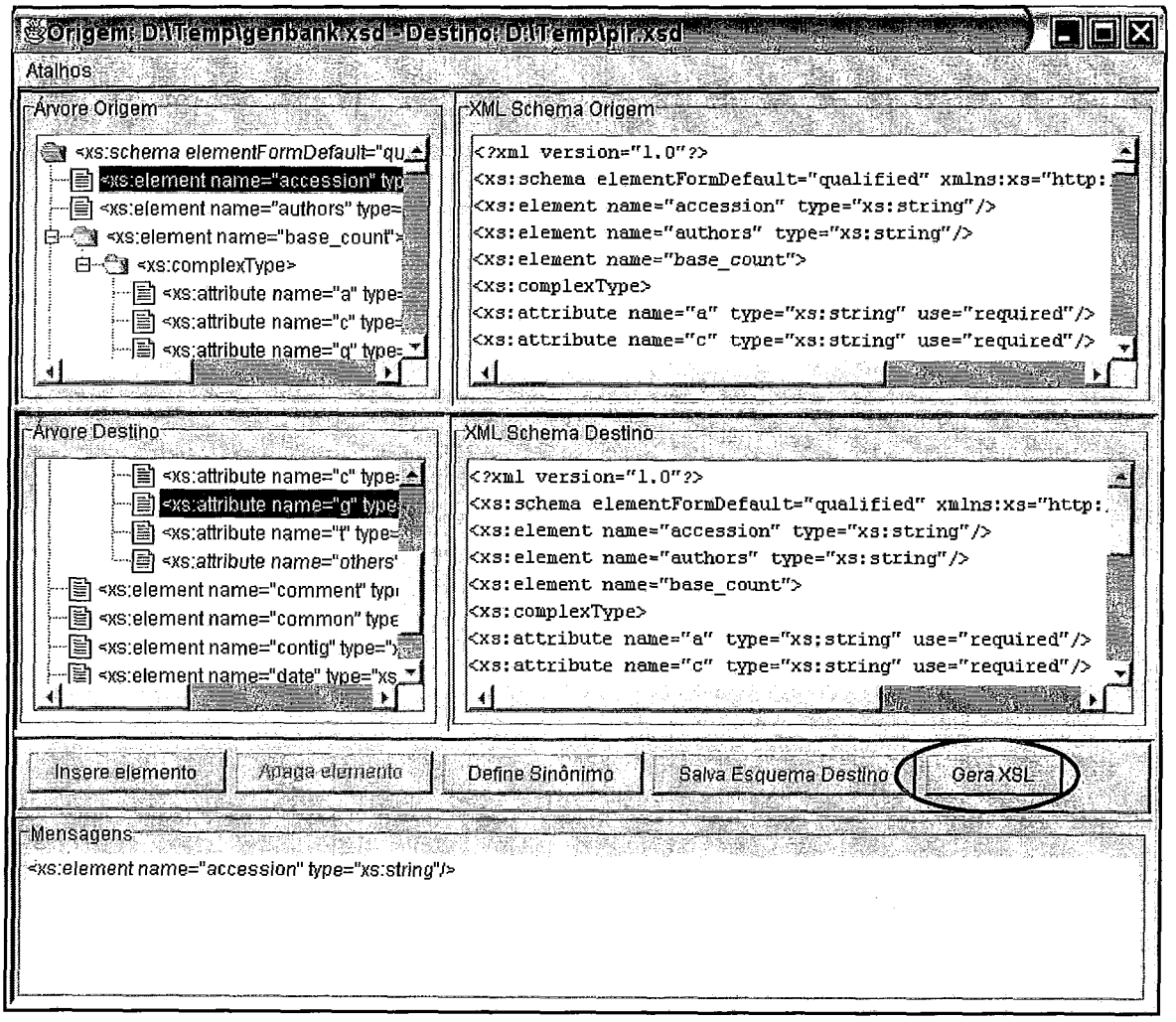


Figura 4-12: Operação "Gera XSL" – Opção XSD

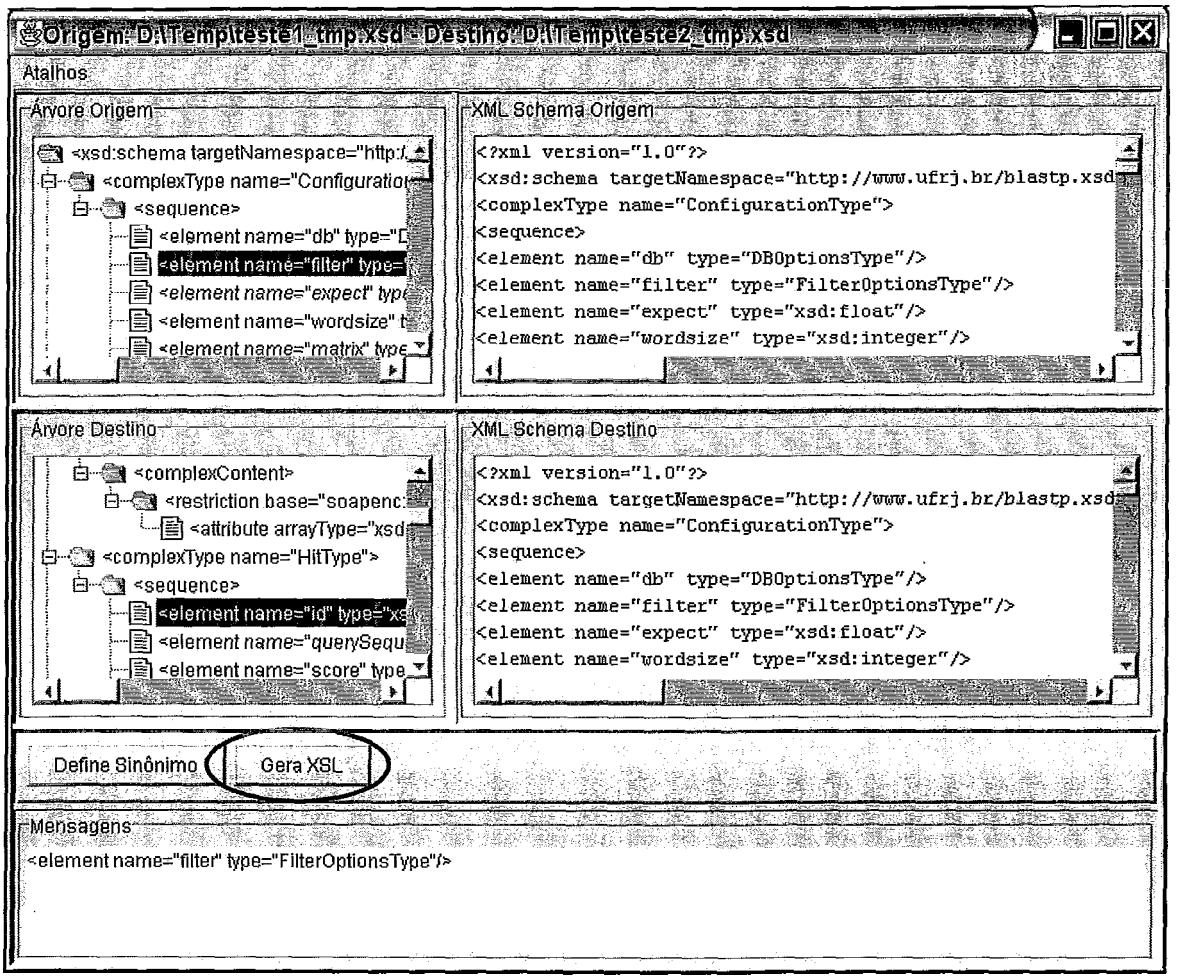


Figura 4-13: Operação "Gera XSL" – Opção WSDL

4.6.6. Outras operações

Algumas operações não foram totalmente exploradas. Problemas como "Namespaces" e "Conflito de nomes" estão fora do escopo desse trabalho. O usuário será responsável por realizar operações que não invalidem o documento XML Schema.

Os esquemas são salvos a cada operação localmente para evitar perda de informações. Antes de salvar o esquema no repositório, este será validado e se apresentar algum erro, uma mensagem será retornada apontando o erro para que a correção seja feita. Apenas documentos XML Schema válidos são salvos no repositório.

4.7. Exemplo de Uso com a Arquitetura SRMW

A fim de avaliar nossa proposta nós a usamos no contexto da arquitetura SRMW, que controla dados biológicos através da Web, usando serviços Web. Essa arquitetura foi explicada anteriormente no capítulo dois. De fato, o módulo da experimentação da SRMW necessita da funcionalidade da ferramenta, a fim de fornecer a conversão dos dados genéticos.

Ao interagir com o módulo da experimentação, o usuário escolhe um conjunto de dados para ser usado como entrada de um programa. Baseado nesta informação, este módulo verifica com a base de dados dos metadados para ver se as categorias de dados, relacionadas à entrada do programa e ao recurso dos dados, são as mesmas ou se já foram mapeadas entre si. Senão, o módulo da experimentação ativa a ferramenta de suporte ao mapeamento para fornecer o mapeamento entre aquelas categorias de dados.

Como exemplo, considere um laboratório genético que trabalhe com a identificação de diversas seqüências genéticas. Geralmente, cada uma (ou um conjunto) destas seqüências é o alvo da pesquisa de um cientista do laboratório. Suponha que este laboratório mantenha o histórico destas seqüências, armazenando-as em um repositório, e que possa exportá-las como um documento XML de acordo com XML Schema específico. Para continuar esta experiência, os cientistas genéticos do laboratório necessitam inferir as estruturas 3D das seqüências do genoma que está sendo estudado. Geralmente, há laboratórios associados especializados nesta inferência que utilizam programas de bioinformática. O Instituto de Biofísica da UFRJ (IBCCF) está desenvolvendo uma metodologia [47] para fazer isso. Eles usam um programa BLAST para processar estas seqüências. Entretanto, este programa tem um formato específico de entrada e como é publicado através da SRMW, como um serviço Web, este formato é descrito em seu arquivo WSDL.

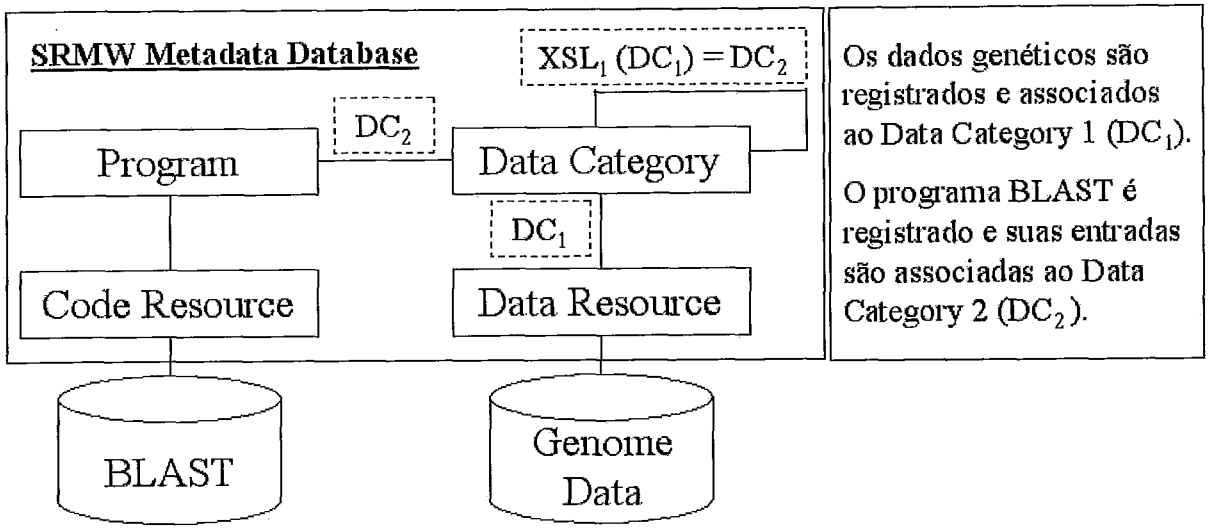


Figura 4-14: Registrando recursos utilizando arquitetura SRMW

Vamos considerar que ambos os recursos estão registrados na arquitetura SRMW (figura 4-14). A seqüência do genoma está registrada como um recurso de dados associado à categoria de dados DC1, enquanto o formato da entrada do BLAST está associado à categoria de dados DC2. Como tanto a seqüência do genoma e o formato da seqüência de entrada programa BLAST estão publicados como serviços Web independentes, DC1 e DC2 são associados a XML Schemas diferentes. Neste exemplo, o usuário necessita executar o BLAST sobre as seqüências do genoma publicadas pelo laboratório genético. Consequentemente, a ferramenta de suporte ao mapeamento é chamada para fornecer o mapeamento entre estes dois esquemas, interagindo com o usuário, e no final gerando um arquivo XSL, que é armazenado então na base de dados dos metadados SRMW. Mais tarde, quando o módulo da experimentação começa o processo da execução, ele recupera o XSL correspondente e aplica ao arquivo XML das seqüências do genoma, transformando automaticamente para o formato de entrada requerido do BLAST.

Exemplo real de utilização da ferramenta descrito passo a passo

O objetivo final desse exemplo é utilizar um arquivo com uma sequência genética no formato GenBank como entrada para um programa de busca de sequências BLAST que requer como entrada arquivos no formato FASTA.

Na figura 4-15 podemos visualizar a sequência no arquivo GenBank no seu formato original. Na figura 4-16 podemos visualizar a mesma sequência no arquivo GenBank no formato XML seguindo o esquema definido no arquivo XML Schema visualizado na figura 4-17.

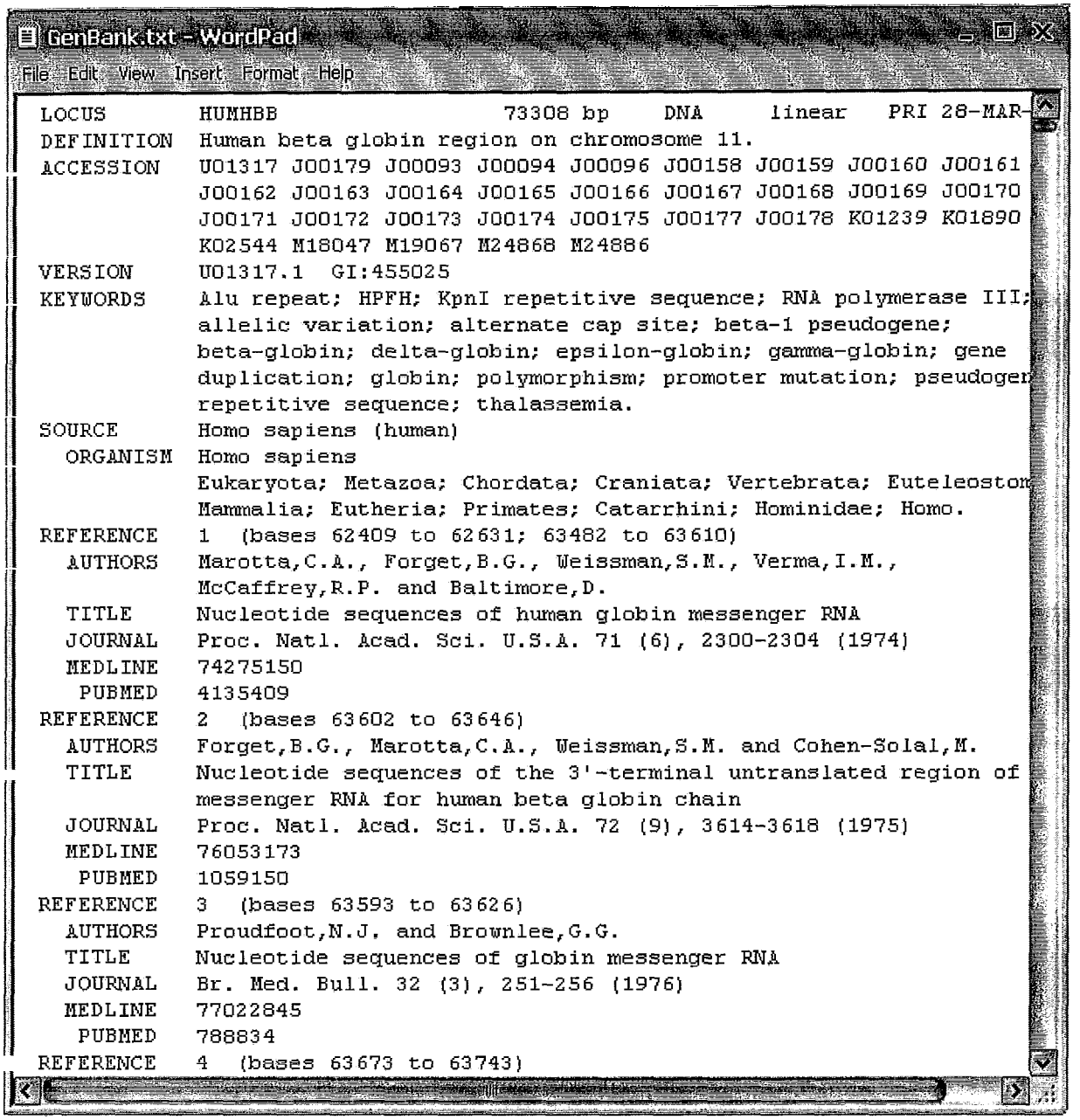


Figura 4-15: Arquivo GenBank no formato original

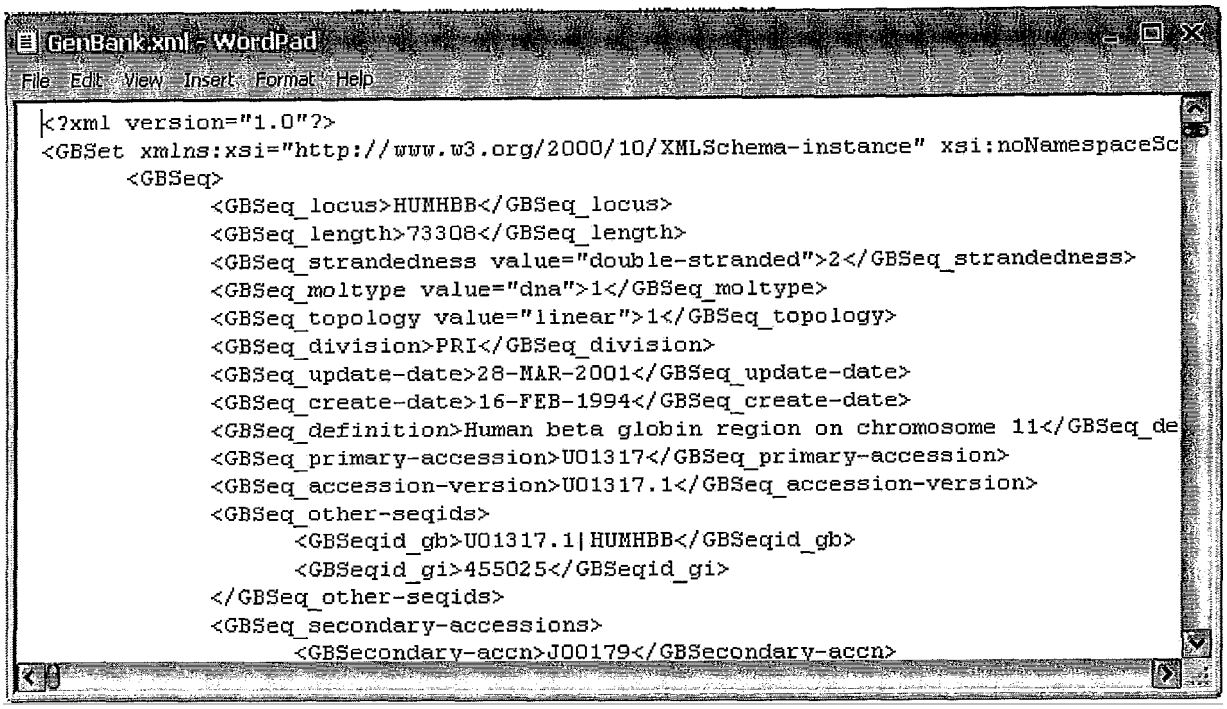


Figura 4-16: Arquivo Genbank no formato XML

Figura 4-17: XML Schema GenBank

Na figura 4-18 podemos visualizar a sequência no arquivo FASTA no seu formato original, ou seja, esse arquivo é o nosso objetivo para utilizar como entrada para o programa BLAST. Na figura 4-19 podemos visualizar a mesma sequência no arquivo FASTA no formato XML seguindo o esquema definido no arquivo XML Schema visualizado na figura 4-20.

Figura 4-18: Arquivo FASTA no formato original

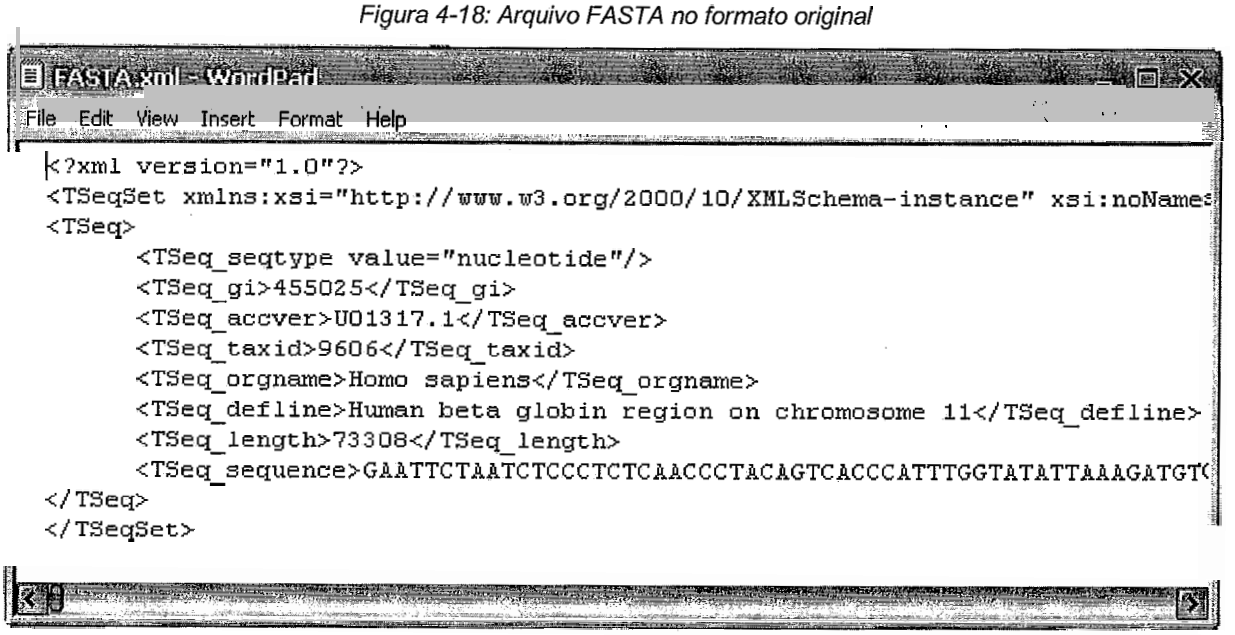


Figura 4-19: Arquivo FASTA no formato XML

Figura 4-20: XML Schema FASTA

O próximo passo é executar a ferramenta informando como entradas:

- Origem: XML Schema GenBank (figura 4-17)
- Destino: XML Schema FASTA (figura 4-20)

Agora será necessário realizar o mapeamento na ferramenta dos elementos GenBank (origem) e FASTA (destino) seguindo a tabela 4-21.

Elemento Origem		**Elemento Destino**
GBSeqid_gi	↔	TSeq_gi
GBSeqid_gb	↔	TSeq_accver
GBSeq_taxon	↔	TSeq_taxid
GBSeq_source	↔	TSeq_orgname
GBSeq_definition	↔	TSeq_defline
GBSeq_length	↔	TSeq_length
GBSeq_sequence	↔	TSeq_sequence

Figura 4-21: Mapeamento GenBank - FASTA

As figuras 4-22 a 4-28 mostram como realizar o mapeamento descrito na tabela 4-21 dentro da ferramenta, através da definição de sinônimos.

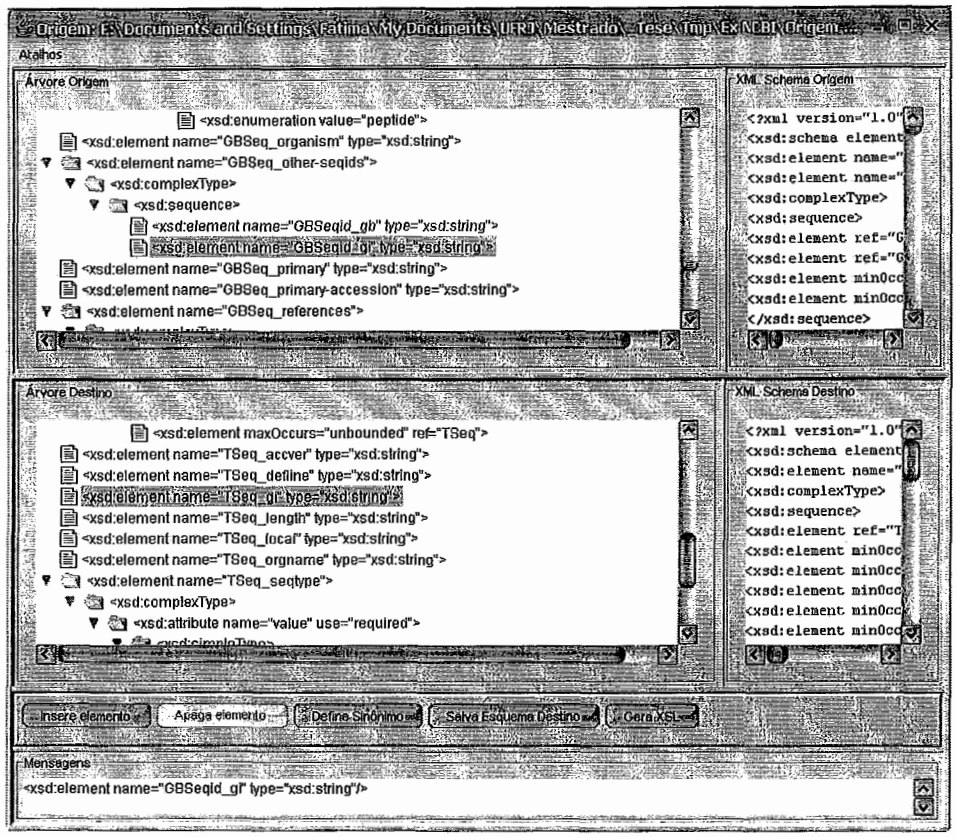


Figura 4-22: Definindo o sinônimo GBSeqid_gi ↔ TSeq_gi

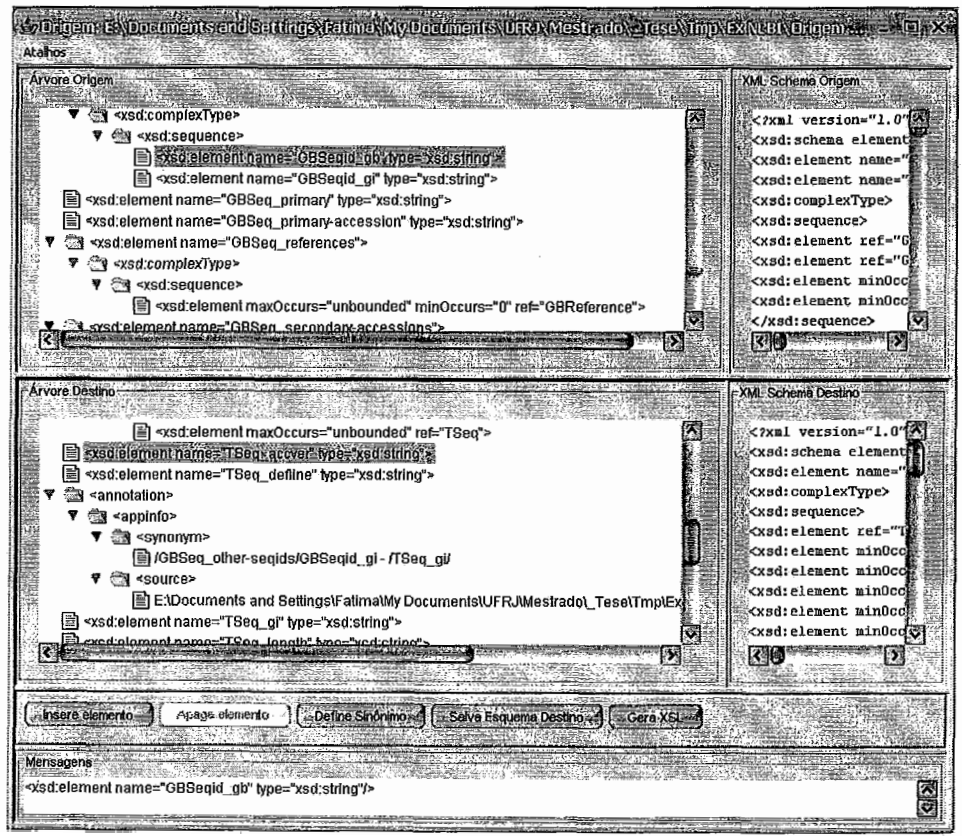


Figura 4-23: Definindo o sinônimo GBSeqid_gb ↔ TSeq_accver

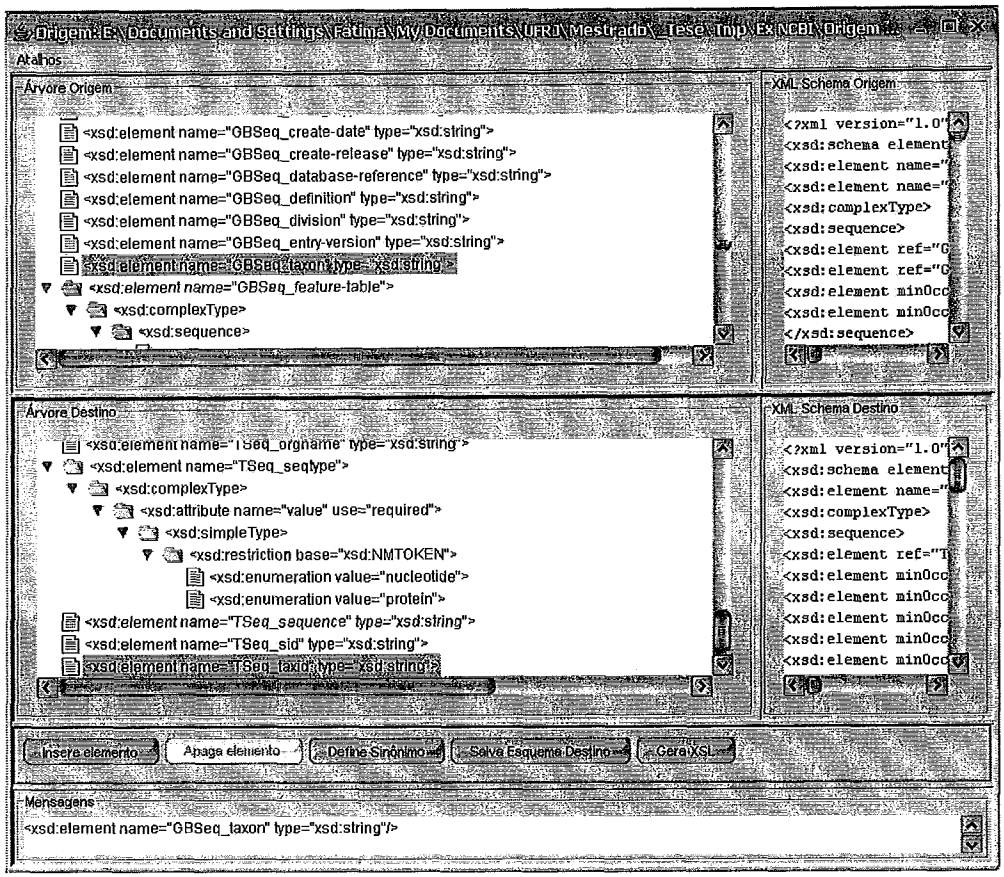


Figura 4-24: Definindo o sinônimo GBSeq_taxon ↔ TSeq_taxid

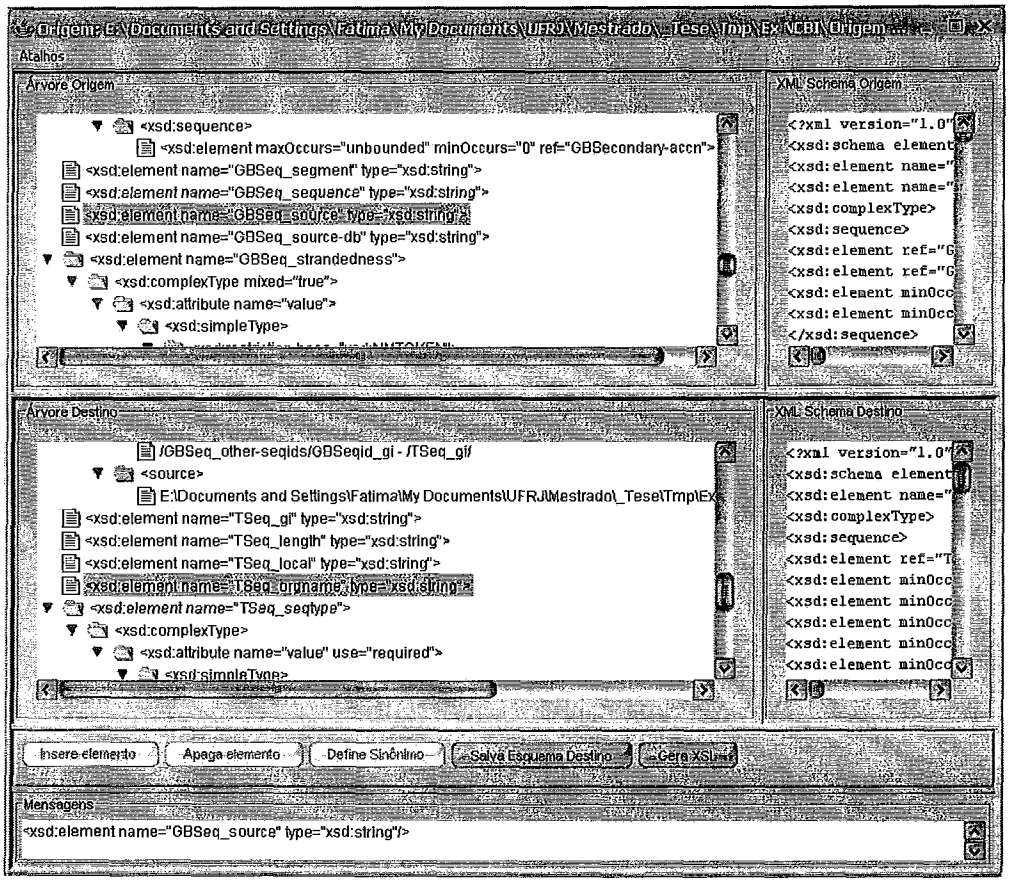


Figura 4-25: Definindo o sinônimo GBSeq_source ↔ TSeq_orgname

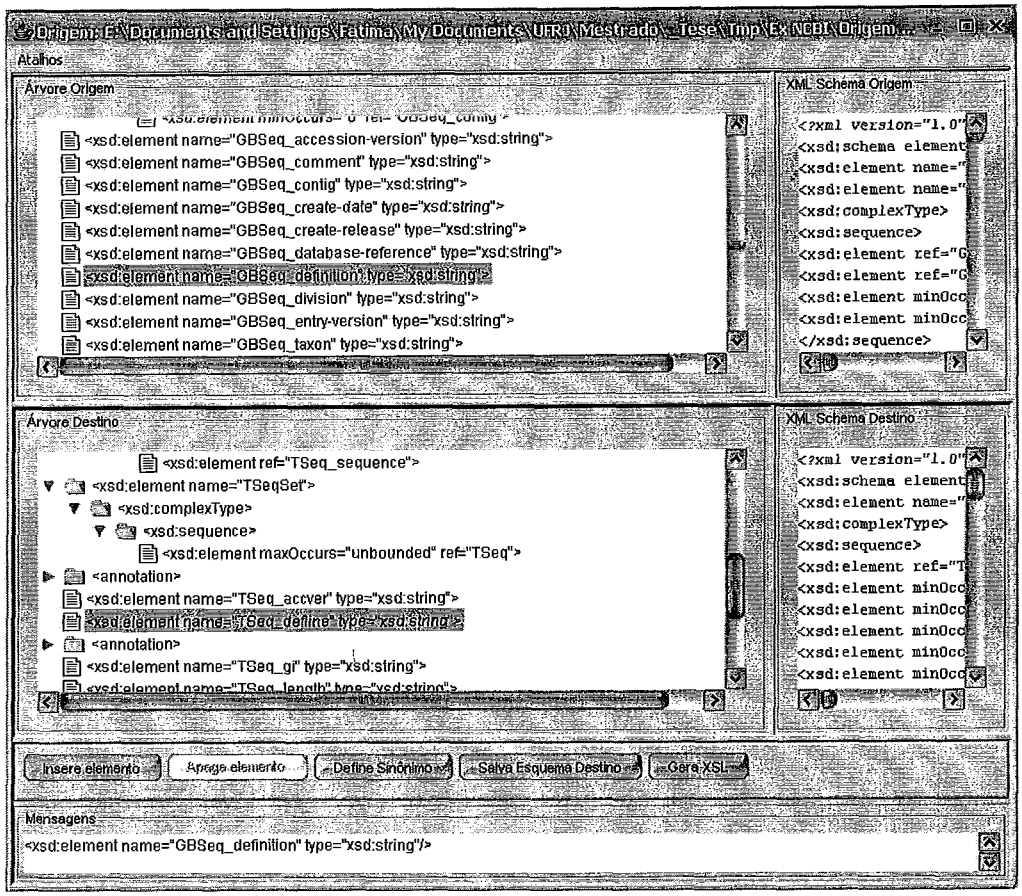


Figura 4-26: Definindo o sinônimo GBSeq_definition ↔ TSeq_defline

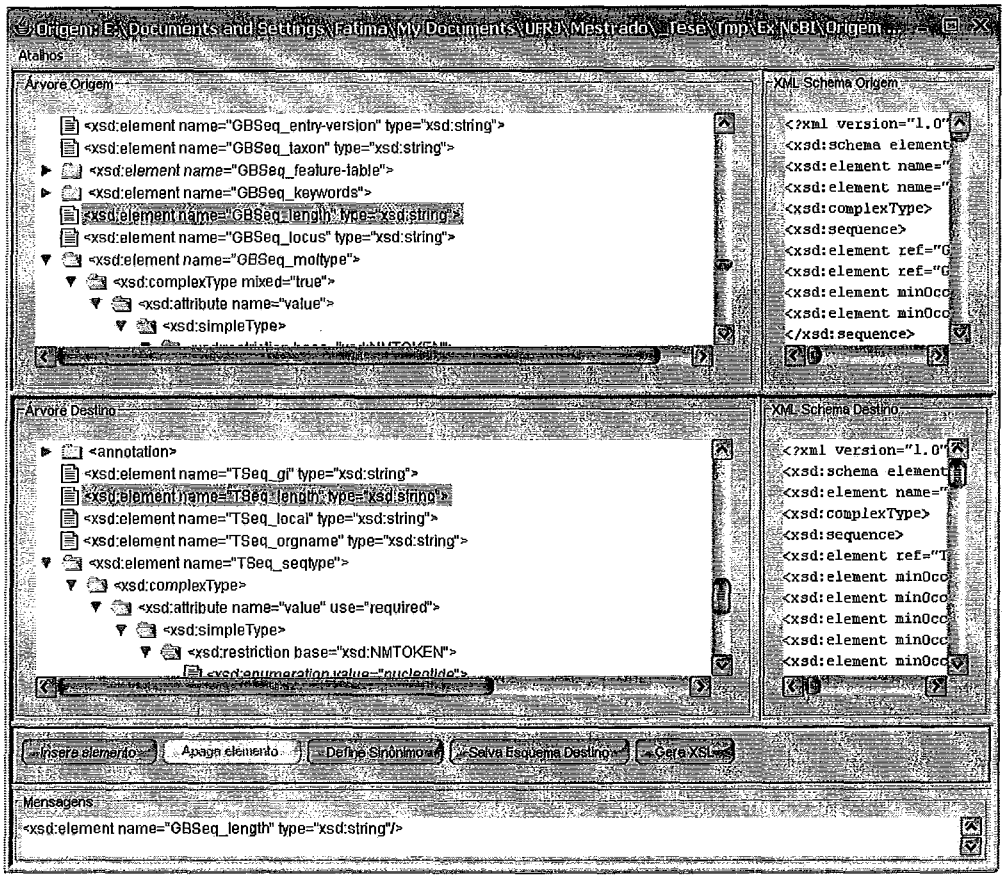


Figura 4-27: Definindo o sinônimo GBSeq_length ↔ TSeq_length

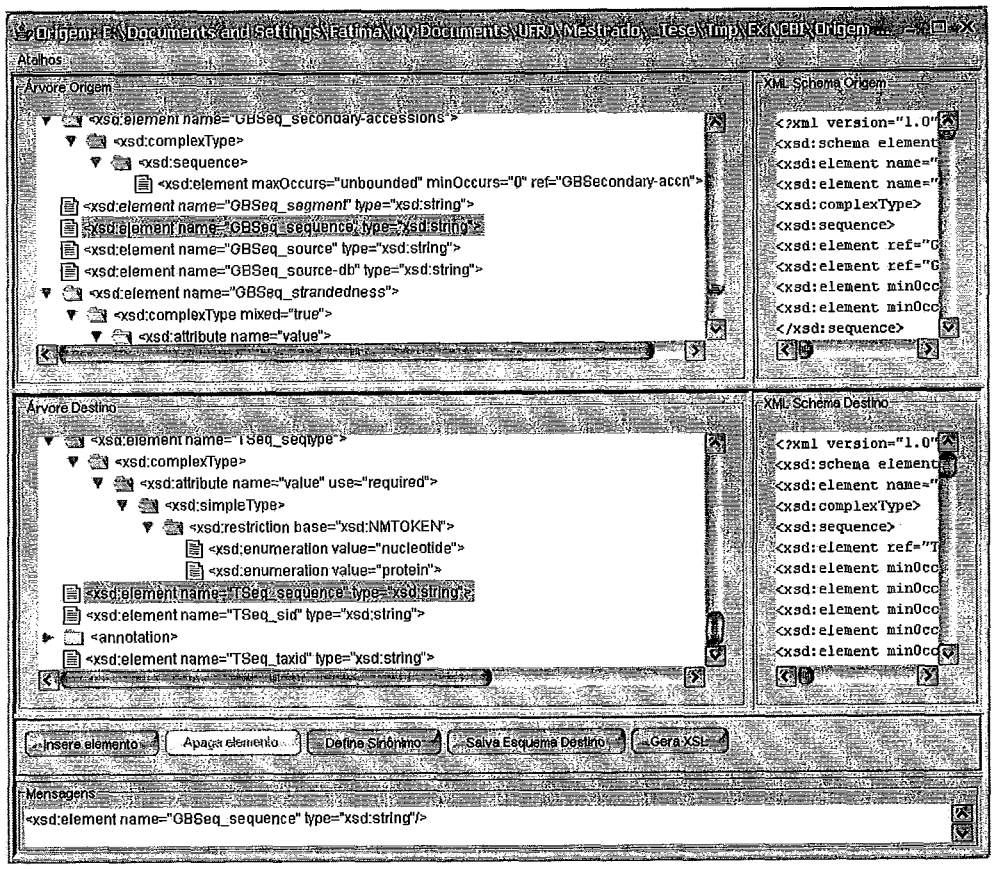


Figura 4-28: Definindo o sinônimo GBSeq_sequence ↔ TSeq_sequence

Depois de definidos todos os sinônimos, podemos finalmente gerar o arquivo XSL que pode ser visualizado na figura 4-27.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
        <TSeqSet>
            <TSeq>
                <xsl:for-each select="GBSeq">
                    <xsl:for-each select="GBSeq_other-seqids">
                        <xsl:for-each select="GBSeqid_gi">
                            <TSeq_gi>
                                <xsl:value-of select="."/>
                            </TSeq_gi>
                        </xsl:for-each>
                    </xsl:for-each>
                </xsl:for-each>
                <xsl:for-each select="GBSeq">
                    <xsl:for-each select="GBSeq_other-seqids">
                        <xsl:for-each select="GBSeqid_gb">
                            <TSeq_accver>
                                <xsl:value-of select="."/>
                            </TSeq_accver>
                        </xsl:for-each>
                    </xsl:for-each>
                </xsl:for-each>
                <xsl:for-each select="GBSeq">
                    <xsl:for-each select="GBSeq_taxon">
                        <TSeq_taxid>
                            <xsl:value-of select="."/>
                        </TSeq_taxid>
                    </xsl:for-each>
                </xsl:for-each>
                <xsl:for-each select="GBSeq">
                    <xsl:for-each select="GBSeq_source">
                        <TSeq_orgname>
                            <xsl:value-of select="."/>
                        </TSeq_orgname>
                    </xsl:for-each>
                </xsl:for-each>
                <xsl:for-each select="GBSeq">
                    <xsl:for-each select="GBSeq_definition">
                        <TSeq_defline>
                            <xsl:value-of select="."/>
                        </TSeq_defline>
                    </xsl:for-each>
                </xsl:for-each>
                <xsl:for-each select="GBSeq">
                    <xsl:for-each select="GBSeq_length">
                        <TSeq_length>
                            <xsl:value-of select="."/>
                        </TSeq_length>
                    </xsl:for-each>
                </xsl:for-each>
                <xsl:for-each select="GBSeq">
                    <xsl:for-each select="GBSeq_sequence">
                        <TSeq_sequence>
                            <xsl:value-of select="."/>
                        </TSeq_sequence>
                    </xsl:for-each>
                </xsl:for-each>
            </TSeq>
        </TSeqSet>
    </xsl:template>
</xsl:stylesheet>
```

Figura 4-29: Arquivo XSL gerado

CAPÍTULO 5. CONCLUSÕES

Mostramos a crescente importância da área de biologia molecular e dos desafios motivados por esse assunto que várias áreas estão enfrentando. Vários grupos de pesquisa estão interessados em soluções para os problemas que envolvem os programas de entrada, acesso, análise e manipulação dos diversos tipos de dados biológicos para os vários sistemas de computadores utilizados nos diferentes laboratórios.

Os dados biológicos são mais complexos quando comparados com a maioria dos outros domínios de aplicações, ao mesmo tempo em que sua quantidade e variedade são muito grandes. Os esquemas das fontes de dados biológicos variam rapidamente e as representações de um mesmo dado por diferentes biólogos podem ser diferentes, ainda que utilizando o mesmo sistema. Além disso, os usuários das informações biológicas geralmente requerem acesso a valores antigos dos dados, particularmente resultados de verificações anteriores.

A necessidade de ter uma maneira autodescritiva e de fácil leitura para a troca de dados genéticos ocorre devido à diversidade dos formatos entre estes dados. Algumas iniciativas de pesquisa adotam um modelo conceitual global para ser usado como um padrão. Outras também consideram usar um modelo único se aproveitando da vantagem do XML que pode ajudar a construção de ferramentas da conversão para os diferentes modelos. O problema dessas propostas está no fato que não há nenhum consenso sobre qual o melhor modelo a ser adotado como um padrão. Além disso, há programas que foram desenvolvidos para tipos específicos de entradas e que teriam que ser reescritos para serem adaptados ao modelo global. Entretanto, mesmo com a adoção de um padrão XML para facilitar a conversão, o problema semântico ainda permanece.

Considerando a necessidade de conviver com os diferentes modelos existentes, surgiram propostas de arquiteturas com o objetivo de solucionar o problema de integração de fontes de dados e programas científicos em seus mais diversos formatos, sem a existência de um esquema único global. Apresentamos duas arquiteturas que motivaram esse trabalho: inicialmente o *framework* do projeto Bio-AXS e posteriormente a arquitetura SRMW.

Nosso trabalho pode ser utilizado como um módulo complementar dessas arquiteturas integradas para trabalhar com as diversas fontes de dados em seus formatos originais e seus *workflows*. Nessas arquiteturas é necessário que o biólogo informe as regras de negócio que possibilitam a integração desejada. Na maioria das vezes, o mapeamento dessas informações não é trivial e realmente depende da interação com o biólogo. Porém, uma vez informado, esse mapeamento poderá ser reaproveitado em utilizações futuras do *workflow*.

Nosso trabalho contribui fornecendo uma ferramenta que dê apoio ao biólogo durante o processo de mapeamento das informações genéticas, interagindo com o usuário de uma maneira flexível e visual. Seu objetivo é facilitar a captura de informações para a integração de diferentes fontes de dados e/ou entradas e saídas de programas. Essa ferramenta pode ser utilizada em conjunto com outras arquiteturas para disponibilizar aos biólogos um ambiente integrado de armazenamento e análise dos dados genéticos independentemente dos formatos em que estão originalmente disponíveis.

Esse módulo foi desenvolvido usando a tecnologia de XML que foi projetada especialmente facilitar a troca de dados. Essa ferramenta explora o poder de representação do XML Schema e o poder de conversão do XSLT.

Apresentamos como nossa ferramenta poderia ser utilizada em conjunto com as arquiteturas de integração mencionadas anteriormente. Citamos também alguns cenários de utilização do módulo de mapeamento de esquemas genéticos que envolvem problemas enfrentados no dia-a-dia dos cientistas. Apresentamos também um exemplo de uma aplicação usando nossa ferramenta no contexto da arquitetura SRMW que utilizamos como validação da nossa proposta.

Nosso trabalho ajuda ao processo de mapeamento que os usuários necessitam tratar durante o encadeamento usual de seus programas ou no armazenamento de dados em algumas bases locais. Uma das vantagens deste módulo é a flexibilidade de se encaixar em diversas arquiteturas e de suportar tanto a conversão de modelos de dados quanto de entradas e saídas de programas. Outra preocupação que tivemos foi projetar um módulo que fosse de fácil utilização, mesmo para usuários que não estivessem familiarizados com as tecnologias envolvidas na área de informática, como é o caso da maioria dos cientistas.

Nossa ferramenta serve como um ponto de partida para outros futuros trabalhos. Atualmente nós temos a opção de definir sinônimos, mas nós planejamos adicionar análise das ontologias como ajuda para encontrar equivalências. Como nossa ferramenta depende principalmente da interação do usuário, um trabalho futuro poderia sugerir os mapeamentos mais utilizados ao usuário baseado nas operações anteriormente realizadas e fazendo uso também das ontologias. Seria interessante pesquisar o registro (*log*) das operações em busca de algumas informações baseadas nos perfis dos usuários, extraindo regras dos mapeamentos mais freqüentes.

CAPÍTULO 6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Human Genome Project: http://www.nhgri.nih.gov/HGP/, 2002.

[2] Human Genome Project Information: http://www.ornl.gov/hgmis/, 2002.

[3] GenBank: http://www.ncbi.nlm.nih.gov/Genbank/index.html, 2002.

[4] AceDB: http://www.acedb.org/, 2002.

[5] Swiss-Prot: http://www.ebi.ac.uk/swissprot, 2002.

[6] S. Lifschitz, A. Miranda, L. Seibel "*A Genome Database Framework*". Database and Expert Systems Applications, 12th International Conference, DEXA 2001, Munique, Alemanha, pp.319-329, Setembro 2001.

[7] National Center for Biotechnology Information (NCBI): http://www.ncbi.nlm.nih.gov, 2002.

[8] XML: http://www.w3.org/XML, 2002.

[9] XML Schema: http://www.w3.org/XML/Schema, 2002.

[10] Oracle 9i: http://www.oracle.com, 2002.

[11] Java JDK 1.3: http://java.sun.com, 2002.

[12] T. Critchlow "*Report on XEWA-00: The XML Enabled Wide-Area Searches for Bioinformatics Workshop*". Sigmod Record, Vol.30, No.1, pp.58-61, Março 2001.

[13] S. Lifschitz, M. Lemos, L. Seibel "*Bancos de Dados de Genoma*". Simpósio Brasileiro de Banco de Dados, SBBD 2000, João Pessoa, Brasil, pp.514-553, Outubro 2000.

[14] NCBI BLAST: http://www.ncbi.nlm.nih.gov/BLAST, 2002.

[15] P. Karp, L. Ruzzo, M. Tompa "*Algoritms in Molecular Biology*", http://www.cs.washington.edu/education/courses/590bi/96wi/, 1996.

[16] V.M. Markowitz, O. Ritter "*Characterizing Heterogeneous Molecular Biology Database Systems*". Journal of Computational Biology, Vol.2, No.4, pp. 547-556, 1995.

[17] S.B. Davidson, C. Overton, P. Buneman "*Challenges in Integrating Biological Data Sources*". Journal of Computational Biology, Vol.2, No.4, pp. 557-572, 1995.

[18] T. Slezak "*What XEWA means to a genomics system builder*". XEWA-00, Texas, Dezembro 2000: http://www-casc.llnl.gov/xewa/xewa-00/.

[19] T. Critchlow "*Strawman Proposal*". XEWA-00, Texas, Dezembro 2000: http://www-casc.llnl.gov/xewa/xewa-00/.

[20] C. Globe "*A Knowledge Representation Briefing*". XEWA-00, Texas, Dezembro 2000: http://www-casc.llnl.gov/xewa/xewa-00/.

[21] "*Resource Description Framework (RDF) Model and Syntax Specification*". Recomendação W3C 19990222, Fevereiro 1999: http://www.w3c.org.

[22] H. Vieira, F. Gonçalves, R. Possato, M. Mattoso *"XML: Definição, Representação e Armazenamento"*, Relatório Técnico, ES-585/02, COPPE/UFRJ, Julho 2002.

[23] S. Davidson *"Tale of Two Cultures: Are there database research issues in bioinformatics"*, 14th International Conference on Scientific and Statistical Database Management, Edimburgo. IEEE Computer Society Press, pp.3-7, Julho 2002.

[24] EBI (EMBL): http://www.ebi.ac.uk/, 2002.

[25] DDBJ: http://www.ddbj.nig.ac.jp/, 2002.

[26] GDB: http://gdbwww.gdb.org/, 2002.

[27] PDB: http://www.rcsb.org/pdb/, 2002.

[28] PIR: http://www-nbrf.georgetown.edu/, 2002.

[29] PROSITE: http://www.expasy.ch/prosite/, 2002.

[30] REBASE: http://rebase.neb.com/rebase/rebase.html, 2002.

[31] Tutoriais sobre os padrões W3C: http://www.w3schools.com/, 2003.

[32] Y. Shohoud *"Building XML Web Services"*, http://www.vbws.com/book/chapter/Ch4Print.aspx, 2002.

[33] PROXIML: http://www.cse.ucsc.edu/~douglas/ proximl, 2003.

[34] PDB and mmCIF: http://www.soe.ucsc.edu/~douglas/proximl/node3.html, 2003.

[35] BIOML: http://www.soe.ucsc.edu/~douglas/proximl/node8.html, 2003.

[36] ProML: http://www.soe.ucsc.edu/~douglas/proximl/node9.html , 2003.

[37] Cavalcanti, M.C. & Mattoso, M. & Campos, M. L. & & Simon, E. & Llirbat, F. *"An Architecture for Managing Distributed Scientific Resources"*. 14th International Conference on Scientific and Statistical Database Management, Edimburgo. IEEE Computer Society Press, pp.47-55, Julho 2002.

[38] Nucleic Acids Research, Vol.28, No.1, 2000.

[39] GAME: http://www.bioxml.org, 2002.

[40] AGAVE: http://www.agavexml.org, 2002.

[41] N. Paton *et al. "Conceptual Modelling of Genomic Information"*. Bioinformatics, Vol.16, No.6, pp.548-557, Junho 2000.

[42] C. Goble, D. De Rouce *"The Grid: An Application of the Semantic Web"*. Sigmod Record, Vol.31, No.4, pp.65-70, Dezembro 2002.

[43] C. Goble *"MyGrid - Directly Supporting the e-Scientist"*: http://www.mygrid.org.uk/, 2003.

[44] T. Critchlow, R. Musick, T. Slezak "*An Overview of Bioinformatics Research at Lawrence Livermore National Laboratory*". DataFoundry Publications, American Biotechnology Laboratory, Vol.18, No.9, Agosto 2000.

[45] M. Cavalcanti. "*Scientific Resources Management: Towards an In Silico Laboratory*", Tese de Doutorado, Relatório Técnico, ES-605/03, COPPE/UFRJ, Junho 2003.

[46] Le Select: http://caravel.inria.fr/Fprototype_LeSelect.html, 2003.

[47] S. Rössle, S. Ribeiro *et al.* "*A Computational Environment Development for the Application of Homology Modeling Methods in Structural Genomic Projects*". Project Poster, IBCCF, UFRJ, Brasil, 2002.

[48] R. McEntire *et al.* "*An Evaluation of Ontology Exchange languages for Bioinformatics*". Proceedings of International Conference on Intelligent Systems for Molecular Biology, ISBM, California. AAAI Press, pp.239-250, 2000.

[49] DataFoundry: www.llnl.gov/CASC/datafoundry, 2003.

[50] Amabis, J. M. Martho. G. R. "*Fundamentos da Biologia Molecular*". Editora Moderna, 1ª edição. São Paulo, 1990.

[51] L. Seibel, M. Lemos, S. Lifschitz. "*Implementation Issues of Bio-AXS: An Object-oriented Framework for Integrating Biological Data and Applications*". Interagency Disability Educational Awareness Showcase, IDEAS 2003, Washington, Estados Unidos, pp.409-413, Novembro 2003.

[52] Santos, C. "*Bancos de Dados Biológicos*". http://www.inf.ufrgs.br/~clesio/cmp151/cmp15120031/BDsBiologicos.pdf, 2003.

[53] Cavalcanti, M., Baião, F., Rössle, S., Bisch, P., Targino, R., Pires, P., Campos, M, Mattoso, M.. "*Structural Genomic Workflows Supported by Web Services*". DEXA 2003, International Workshop on Biological Data Management, BIDM 2003, IEEE CS Press, Praga, República Checa, pp.45-49, Setembro 2003.

[54] BDGP: http://fruitfly.org, 2004.

[55] E-Science DTI: http://www.escience-grid.org.uk/index.htm, 2004.

[56] Goble, C. *et al.* "*On the Use of Agents in BioInformatics Grid*". CCGRID 2003, pp. 653-668.

[57] Goble, C. "*The Grid Needs You! Enlist Now*". CoopIS/DOA/ODBASE 2003, pp. 589-600

[58] Stevens, R., Robinson, A., Goble, C. "*myGrid: personalised bioinformatics on the information grid*". ISMB (Supplement of Bioinformatics) 2003, pp. 302-304

[59] L. Seibel, M. Lemos, S. Lifschitz. "*A Conceptual Model for Molecular Biology Information*", Second Brazilian Workshop on Bioinformatics, WOB'03, Macaé, Brasil, Dezembro 2003.

[60] H. Vieira, F. Gonçalves, M. Mattoso *"XQuery, XML Schema e XSL"*, Publicações Técnicas 2/2002, COPPE/UFRJ, Julho 2002.

[61] F. Gonçalves, M. Cavalcanti, M. Mattoso, "*Genetic Schema Mapping Support*", Second Brazilian Workshop on Bioinformatics, WOB'03, Macaé, Brasil, Dezembro 2003.

[62] M. Cavalcanti, R. Targino, F. Baião, S. Rossle, P. Bisch, P. Pires, M. Campos, M. Mattoso. "*Managing Structural Genomic Workflows using Web Services*" Data & Knowledge. Engineering Journal., a ser publicado, 2004.

[63] F. Teixeira, M. Cavalcanti, F. Baião, L. Meyer, S. Rossle, P. Bisch, P. Pires, M. Mattoso. "*Data Management via Web Services in Bioinformatics Workflows*", Second Brazilian Workshop on Bioinformatics, WOB'03, Macaé, Brasil, Dezembro 2003.

[64] M. Cornell, N. Paton, C. Hedeler, P. Kirby, D. Delneri, A. Hayes, S. Oliver. "*GIMS: An Integrated Data Storage and Analysis Environment for Genomic and Functional Data*", Yeast, Vol. 20, No. 15, pp.1291-1306, 2003.

[65] M. Cornell, N. Paton, S. Wu, C. Goble, C. Miller, P. Kirby, K. Eilbeck, A. Brass, A. Hayes, S. Oliver. "*GIMS - A Data Warehouse for Storage and Analysis of Genome Sequence and Functional Data*", 2nd IEEE International Symposium on Bioinformatics and Bioengineering (BIBE), IEEE CS Press, pp.15-22, 2001.

[66] S. Rössle, S. Ribeiro, *et al.* "*A Computational Environment Development for the Application of Homology Modeling Methods in Structural Genomic Projects*", Project Poster, IBCCF, UFRJ, Brasil, 2002.

[67] M. Ozsu, P. Valduriez. "*Principles of Distributed Database Systems*", Prentice Hall, 1999.

[68] G. Wiederhold *et al.*. "*Medical Informatics: Computer Applications in Health Care and Biomedicine*". Addison-Wesley, second edition, Springer Verlag, 2000.

[69] G. Wiederhold. "*Intelligent Integration of Information*". Kluwer Academic Publishers, Boston, 1996.

[70] M. Mattoso, M. Cavalcanti, R. Souza, H. Teixeira, L. Azevedo, C. Marques, R. Monteiro, F. Gonçalves, C. Werner. "*Gerência de Documentos XML no GOA*", XVI Simpósio Brasileiro de Engenharia de Software, Ferramentas, pp.402-407, Gramado, Brasil, Outubro 2002.

APÊNDICE A – WSDL

WSDL significa **Web Services Description Language**. WSDL é um documento escrito em XML que descreve um Web Service. Serve para especificar uma localização de um serviço e as operações ou métodos que o serviço possui.

WSDL ainda não é um padrão W3C, é uma proposta da Ariba, IBM e Microsoft para descrever serviços sobre protocolos XML. Deve se tornar um padrão W3C até o final do ano de 2003.

➢ Estrutura WSDL

Um documento WSDL define um Web Service usando os elementos a seguir:

Elemento	Define
<portType>	As operações realizadas pelo Web Service
<message>	As mensagens usadas pelo Web Service
<types>	Os tipos de dados usados pelo Web Service
<binding>	Os protocolos de comunicação usados pelo Web Service

Exemplo da estrutura principal de um documento WSDL:

```
<definitions>
<types>
  definição de tipos.....
</types>
<message>
  definição da mensagem....
</message>
<portType>
  definição de uma porta.....
</portType>
<binding>
  definição de um binding....
</binding>
</definitions>
```

Um documento WSDL pode conter também outros elementos como elementos estendidos e um elemento de serviço que pode torna possível a definição de vários Web Services em um único documento WSDL.

- Portas WSDL

O elemento **<portType>** é o elemento mais importante do WSDL. Ele define um Web Service, as operações que ele pode realizar e as mensagens que estão envolvidas. Esse elemento pode ser comparado à uma biblioteca de funções (ou um módulo ou uma classe) nas linguagens de programação tradicionais.

- Mensagens WSDL

O elemento **<message>** define os elementos de dados de uma operação. Cada mensagem pode consistir de uma ou mais partes. As partes podem ser comparadas à parâmetros da chamada de uma função nas linguagens de programação tradicionais.

- Tipos WSDL

O elemento **<types>** define os tipos de dados usados pelo Web Service. Para se obter uma plataforma o mais neutra possível, WSDL usa a sintaxe do XML Schema para definir os tipos de dados.

- WSDL Bindings

O elemento **<binding>** define o formato da mensagem e os detalhes de protocolo de cada porta.

➢ Exemplo WSDL

A seguir temos uma fração simplificada de um documento WSDL:

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
 <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
 </operation>
</portType>
```

Nesse exemplo o elemento **portType** define "glossaryTerms" como o nome de uma porta (**port**) e "getTerm" como o nome da operação (**operation**). A operação "getTerm" tem uma mensagem de entrada (**input message**) chamada "getTermRequest" e uma mensagem de saída (**output message**) chamada "getTermResponse". O elemento **message** define as partes de uma mensagem e os tipos de dados associados.

Comparando com a programação tradicional, "glossaryTerms" é uma biblioteca de funções, "getTerm" é uma função com "getTermRequest" como um parâmetro de entrada e "getTermResponse" como um parâmetro de retorno.

➢ Tipos de operações

Existem quatro tipos de operações:

Tipo	Definição
One-way	A operação pode receber uma mensagem, mas não vai retornar uma resposta.
Request-response	A operação pode receber uma mensagem e vai retornar uma resposta.
Solicit-response	A operação pode mandar uma mensagem e vai esperar uma resposta.
Notification	A operação pode mandar uma mensagem, mas não vai esperar uma resposta.

- Exemplo de operação one-way:

```
<message name="newTermValues">
  <part name="term" type="xs:string"/>
  <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
  <operation name="setTerm">
    <input name="newTerm" message="newTermValues"/>
  </operation>
</portType >
```

Nesse exemplo a porta "glossaryTerms" define uma operação one-way chamada "setTerm" que permite a entrada de novos termos de glossário usando a mensagem "newTermValues" com os parâmetros de entrada "term" e "value". Entretanto, nenhuma saída é definida para essa operação.

- Exemplo de operação request-response:

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>
<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
 <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
 </operation>
</portType>
```

Nesse exemplo a porta "glossaryTerms" define uma operação request-response chamada "getTerm" que requer uma mensagem de entrada "getTermRequest" com um parâmetro chamado "term" e vai retornar uma mensagem de saída chamada "getTermResponse" com um parâmetro chamado "value".

➢ UDDI

UDDI (Universal Description, Discovery and Integration) é um serviço de diretório onde pode se registrar e procurar por Web Services descrito em WSDL. UDDI é um framework de plataforma independente para descrição de serviços, descoberta de negócios e integração de serviços de negócios utilizando a Internet. UDDI se comunica via SOAP e foi construído sobre a plataforma NET da Microsoft.

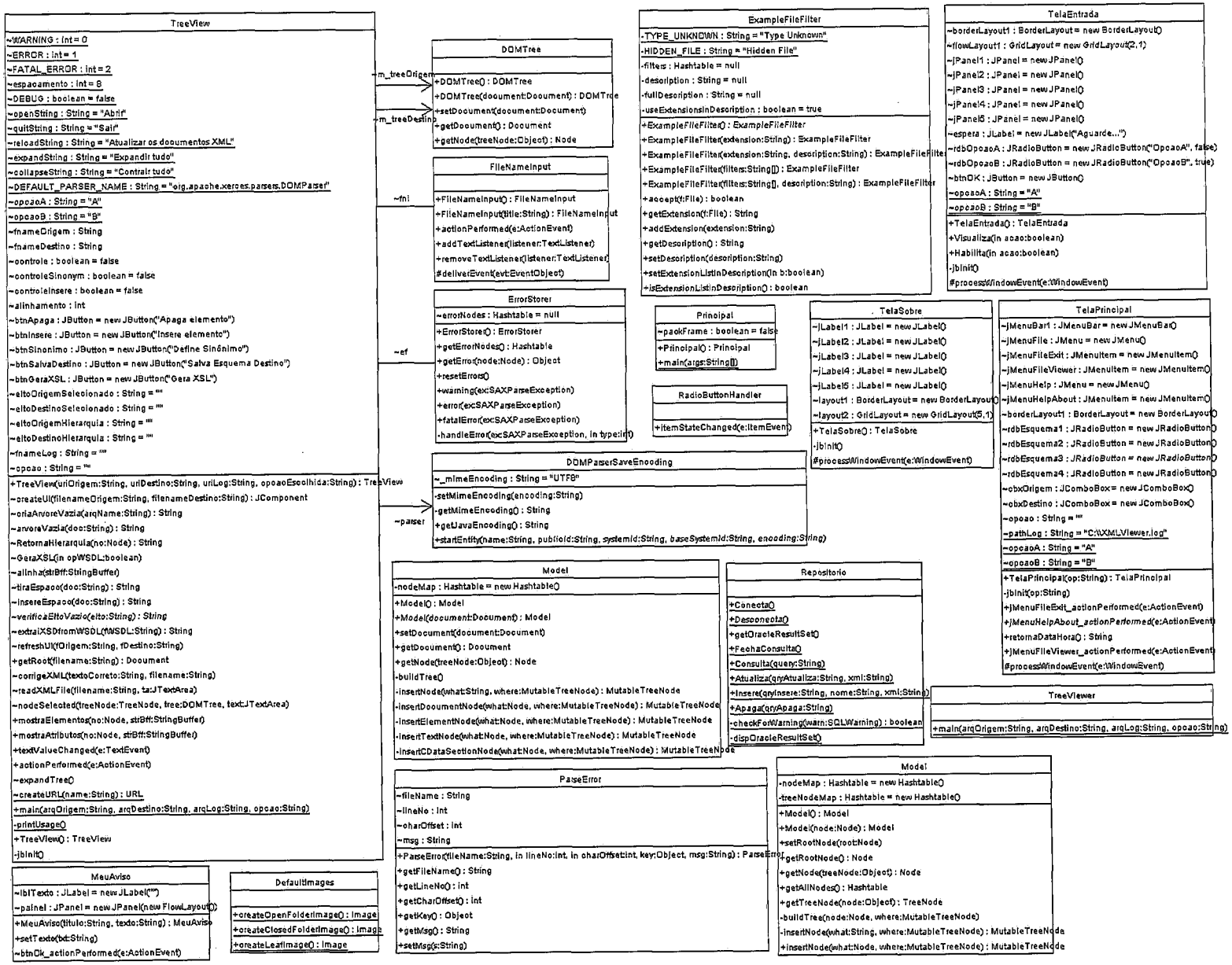
➢ Sintaxe WSDL

A sintaxe completa do WSDL 1.2 como descrita pelo "W3C Working Draft" está listada a seguir [31]:

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri">
  <import namespace="uri" location="uri"/> *
  <wsdl:documentation .... /> ?
  <wsdl:types> ?
    <wsdl:documentation .... /> ?
    <xsd:schema .... /> *
  </wsdl:types>
  <wsdl:message name="ncname"> *
    <wsdl:documentation .... /> ?
    <part name="ncname" element="qname"? type="qname"?/> *
  </wsdl:message>
  <wsdl:portType name="ncname"> *
    <wsdl:documentation .... /> ?
    <wsdl:operation name="ncname"> *
      <wsdl:documentation .... /> ?
      <wsdl:input message="qname"> ?
        <wsdl:documentation .... /> ?
      </wsdl:input>
      <wsdl:output message="qname"> ?
        <wsdl:documentation .... /> ?
      </wsdl:output>
      <wsdl:fault name="ncname" message="qname"> *
        <wsdl:documentation .... /> ?
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:serviceType name="ncname"> *
    <wsdl:portType name="qname"/> +
  </wsdl:serviceType>
  <wsdl:binding name="ncname" type="qname"> *
    <wsdl:documentation .... /> ?
    <-- binding details --> *
    <wsdl:operation name="ncname"> *
      <wsdl:documentation .... /> ?
      <-- binding details --> *
      <wsdl:input> ?
        <wsdl:documentation .... /> ?
        <-- binding details -->
      </wsdl:input>
      <wsdl:output> ?
        <wsdl:documentation .... /> ?
        <-- binding details --> *
      </wsdl:output>
      <wsdl:fault name="ncname"> *
        <wsdl:documentation .... /> ?
        <-- binding details --> *
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="ncname" serviceType="qname">
*
    <wsdl:documentation .... /> ?
    <wsdl:port name="ncname" binding="qname"> *
      <wsdl:documentation .... /> ?
      <-- address details -->
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

APÊNDICE B – Diagramas da Ferramenta

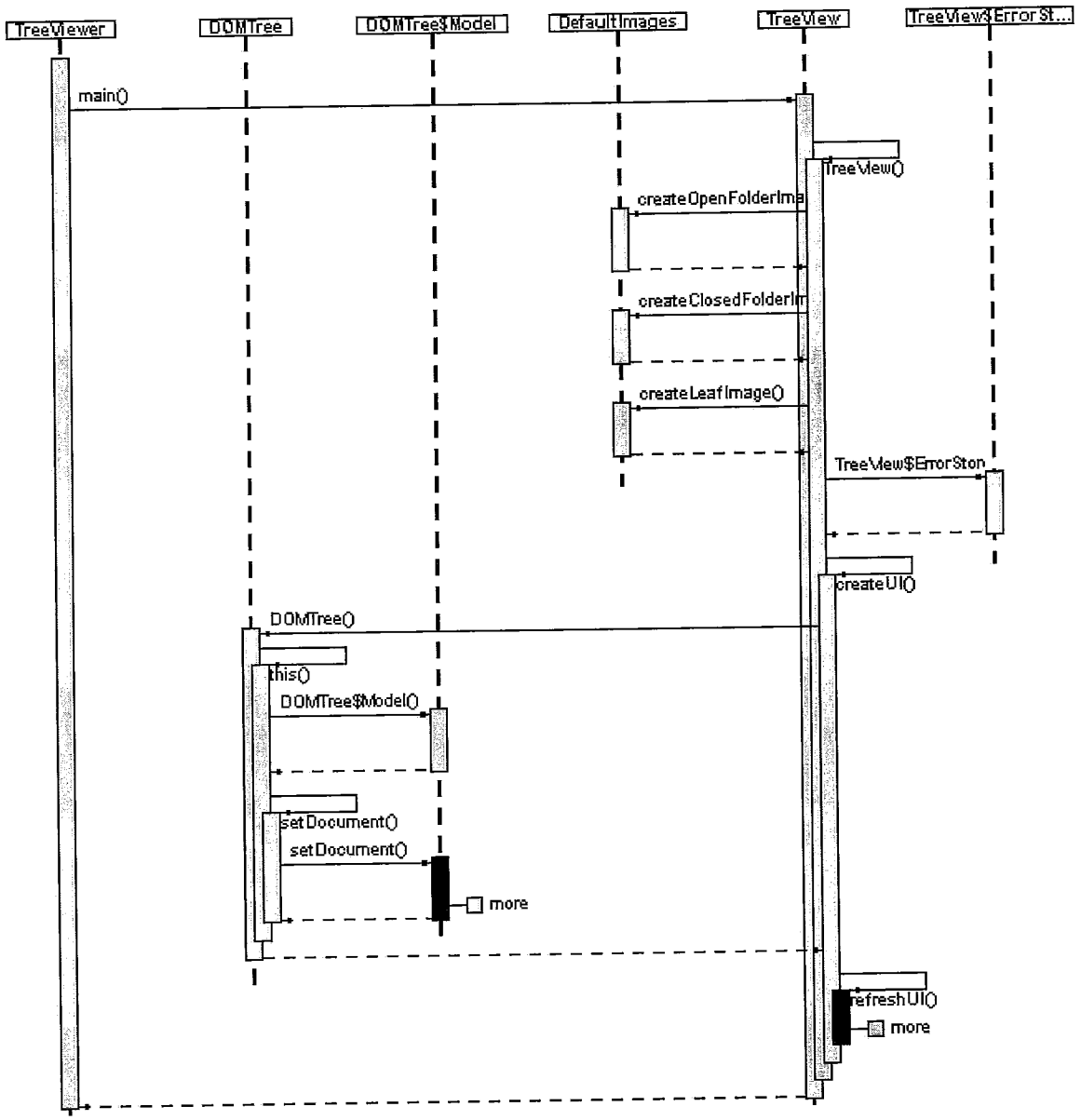
➢ Diagrama de Classe



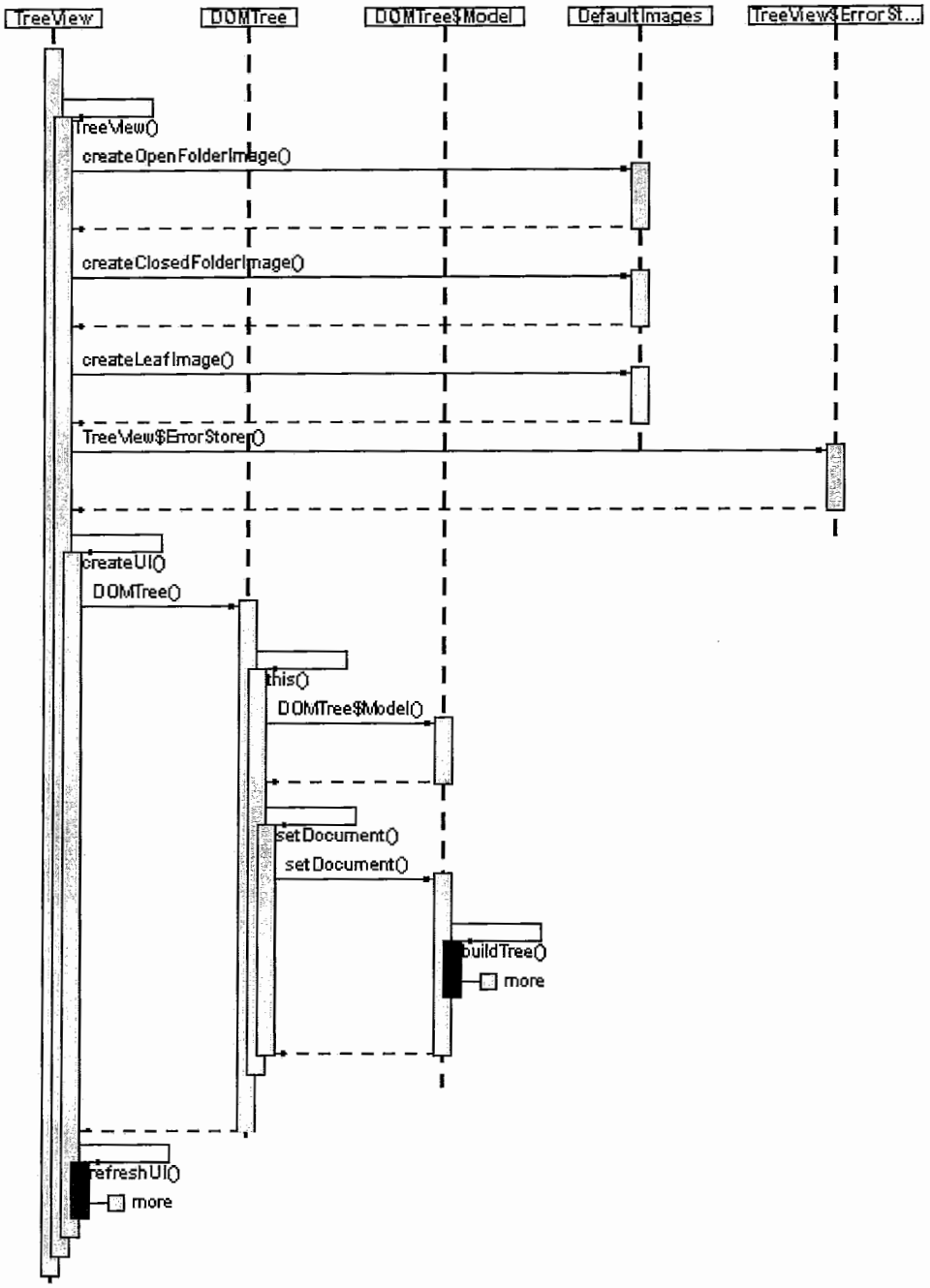
➢ Diagramas de Seqüência

Nas próximas páginas foram apresentados apenas alguns diagramas de seqüência do módulo proposto relacionados aos principais métodos envolvidos.

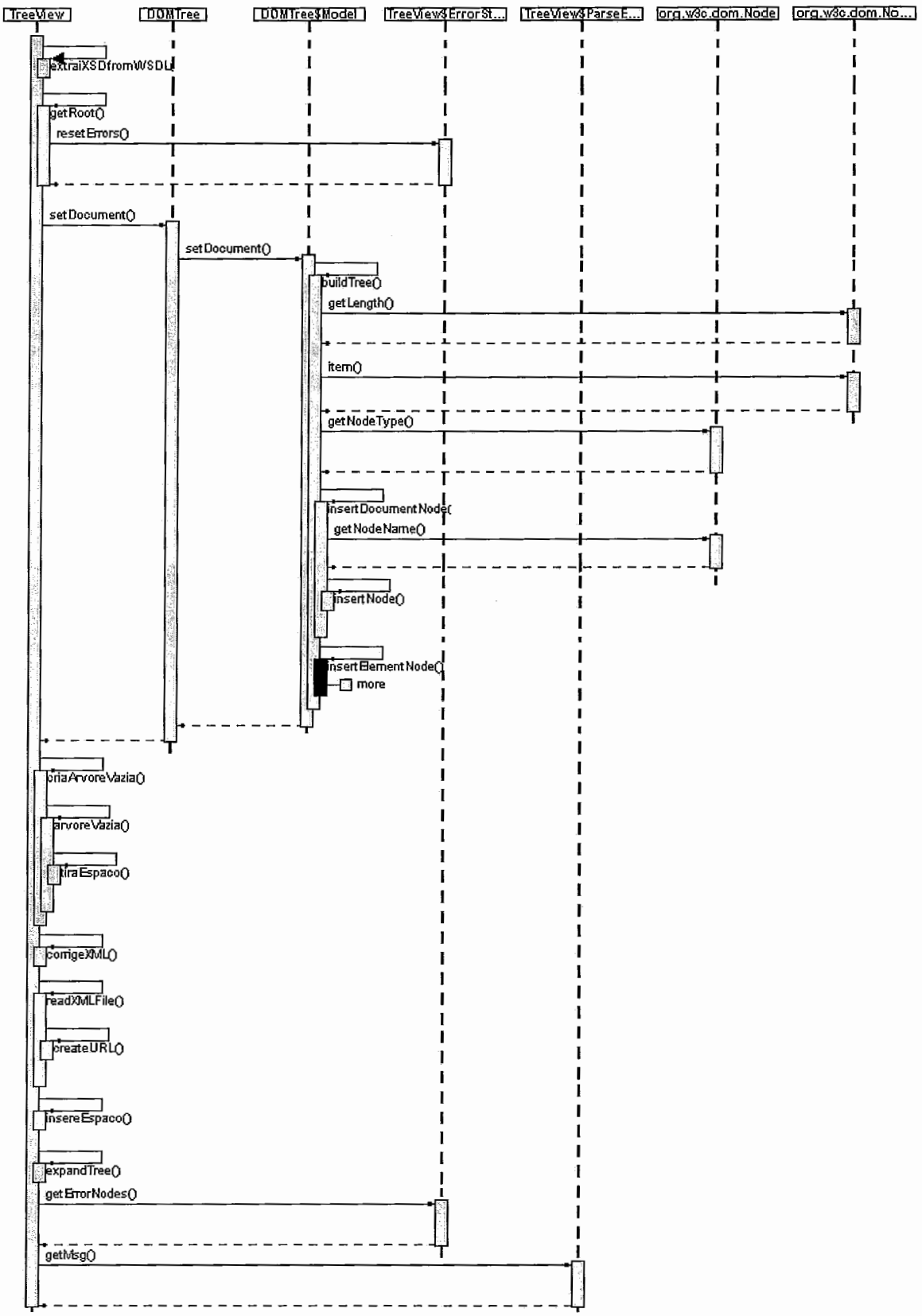
- Classe TreeViewer – Método main



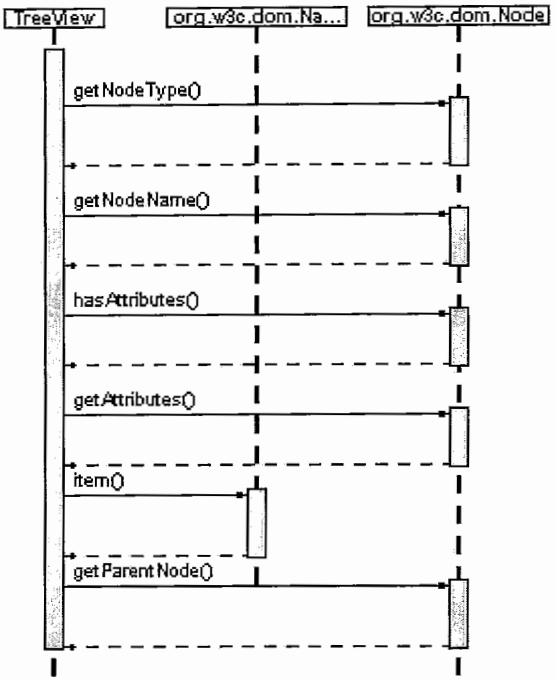
- Classe TreeView – Método main



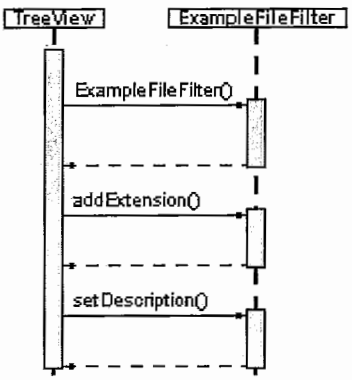
- Classe TreeView – Método refreshUI



- Classe TreeView – Método retornaHierarquia



- Classe TreeView – Método geraXSL



- Classe TreeView – Método nodeSelected