UNIVERSIDADE FEDERAL DA BAHIA

INSTITUTO DE MATEMÁTICA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Ivan Carmo da Rocha Neto

Desenvolvimento de um Ambiente Integrado de apoio à Análise Filogenética através de Redes Complexas

Salvador

2011.2

Ivan Carmo da Rocha Neto

Desenvolvimento de um Ambiente Integrado de apoio à Análise Filogenética através de Redes Complexas

Monografia apresentada ao Curso de graduação em Ciência da Computação, Departamento de Ciência da Computação, Instituto de Matemática, Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Profº . Antonio Lopes Apolinário Junior.
Co-orientador: Profº . Roberto Fernandes Silva Andrade.

Salvador

2011.2

RESUMO

O método de análise filogenética através de redes complexas parte de princípios teóricos da Física Estatística e utiliza técnicas computacionais baseadas em Teoria dos Grafos. Essa abordagem se mostra como uma alternativa viável aos pesquisadores, na medida em que traz resultados comparáveis aos de outros métodos. Neste trabalho de análise e desenvolvimento são organizados os dados, informações e programas desenvolvidos pelo grupo de pesquisa FESC (Física Estatística e Sistemas Complexos - Instituto de Física - UFBA), e é projetado um *software* visando a facilidade de uso por biólogos, físicos e bioinformáticos como uma importante alternativa às ferramentas existentes para análise filogenética. Os artefatos originais foram desenvolvidos com falta de integração e uma forma de organização que dificulta a utilização por parte dos pesquisadores. O produto deste trabalho é um ambiente integrado de controle e gerenciamento de artefatos, com a utilização de persistência e modelado de forma que permita extensibilidade.

Palavras-chave: análise filogenética, engenharia de software, reengenharia, modelagem, desenvolvimento.

ABSTRACT

The method of phylogenetic analysis through complex networks comes from the theoretical principles of Statistical Physics and uses computational techniques based on Graph Theory. There is an evidence that this approach is a viable alternative to the researchers, because it has results that are comparable to other methods. In this work of analysis and development data, information and programs developed by FESC (Group of Statistical Physics and Complex Systems - Physics Institute - UFBA) are organized, and it is designed a software that is easily useful by biologists, physicists and bioinformaticians as an important alternative of existing tools for phylogenetic analysis. The original artifacts were developed with a lack of integration and form of organization that makes it difficult to use by researchers. The product of this work is an integrated control and management of artifacts, with the use of persistence and modeled to allow extensibility.

Keywords: phylogenetic analysis, software engineering, reengineering, modeling, development

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AIAF	Ambiente Integrado de apoio à Análise Filogenética através de Redes Complexas,	p. 16
BLAST	*Basic Alignment Search Tool*,	p. 12
FESC	Grupo de pesquisa em Física Estatística e Sistemas Complexos - Instituto de Física - UFBA,	p. 9
GUI	*Graphical User Interface*,	p. 23
NCBI	*National Center for Biotechnology Information*,	p. 11
SSH	*Secure Shell*,	p. 39

SUMÁRIO

1 INTRODUÇÃO

A crescente evolução dos computadores e sua capacidade de processamento vem influenciando diretamente as mais diversas áreas do conhecimento. A Biologia tem aproveitado o crescimento do poder computacional para realizar análises cada vez mais poderosas sobre espécies, organismos e sua carga genética. A possibilidade de comparação de grandes sequências de DNA em um curto intervalo de tempo possibilita resultados impossíveis de serem obtidos através de uma comparação manual.

A Bioinformática é hoje uma das áreas multidisciplinares mais promissoras, e a comparação de sequências proteicas permite um impulso na área de Análise Filogenética, onde diversos métodos tem surgido. Outras áreas do conhecimento que têm se beneficiado bastante com a crescente evolução computacional são a Física e a Matemática que, assim como a Biologia, podem atingir resultados rapidamente antes impossíveis, e acabam gerando um leque de opções para pesquisa em computação aplicada.

Hoje em dia a demanda por profissionais e pesquisadores de Ciência da Computação nestas áreas tem se mostrado muito grande, e a junção de duas ou mais delas em um contexto multidisciplinar tem guiado as pesquisas a resultados muito interessantes, além de gerar produtos e também ideias para novos projetos de pesquisa. Em um contexto de Física Estatística e Sistemas Complexos, aliando-se conceitos matemáticos da Teoria dos Grafos, foi possível o desenvolvimento de um método para Análise Filogenética, que levou a resultados comparáveis com os de outros métodos existentes (ANDRADE et al., 2011).

Nos três anos de pesquisa e desenvolvimento do método, houve uma maior preocupação com resultados, o que fez com que o processo de desenvolvimento de artefatos computacionais para que se chegasse a eles não seguisse um metodologia de desenvolvimento baseada em conceitos de Engenharia de Software. Como resultado desse processo de construção de software sem metodologia, o ambiente computacional gerado necessita de uma curva de aprendizado grande para a sua execução, além de cuidados com relação à organização e nomenclatura de arquivos, configurações, formatação de dados de entrada e saída para o uso em diferentes programas. A partir do momento em que o método está concluído, testado e comparado com

outros métodos, se faz necessário sua organização para diminuir esta curva de aprendizado. A aplicação de uma metodologia de Engenharia de Software ao sistema atual se dá a partir de (i) concepção de uma arquitetura que permita uma separação em camadas de interação com o usuário, gerenciamento de e controle, e persistência; (ii) controle de execução do processo; (iii) possibilidade de extensibilidade; (iv) utilização de código legado.

1.1 MOTIVAÇÃO

A proposta deste trabalho surgiu a partir do início de 2007, quando o grupo de pesquisa Física Estatística e Sistemas Complexos (FESC), do Instituto de Física da Universidade Federal da Bahia, resolveu utilizar sua experiência com Redes Complexas aplicando-a a Análise Filogenética. Os resultados obtidos com a aplicação do método a dados biológicos abrem espaço para a utilização do mesmo como uma alternativa aos métodos já existentes no contexto da pesquisa em Análise Filogenética. A partir do momento em que esta alternativa se torna viável, na medida em que traz resultados válidos e comparáveis aos dos métodos já conhecidos, faz-se necessária a sua organização para que seja utilizada em larga escala, além de sua extensão, resultando em um ambiente em que haja a facilidade de uso, desenvolvimento, manutenção e extensão, além da possibilidade de compartilhamento de resultados entre os diferentes métodos existentes.

1.2 OBJETIVOS

O objetivo deste trabalho é aplicar metodologias baseadas em Engenharia de Software de forma a desenvolver um ambiente integrado que agrupe um conjunto de ferramentas e códigos legados desenvolvidos pelo grupo de pesquisa FESC (Física Estatística e Sistemas Complexos), utilizando-se de pouca metodologia. Em seguida, um levantamento do que era necessário para que se pudesse organizar os programas e *scripts* utilizados objetivando melhorar sua execução e experiência de utilização (a serem comprovadas após validação) por pesquisadores que detêm de pouco ou quase nenhum conhecimento de desenvolvimento de software e ambientes Linux, além de permitir sua evolução e extensibilidade.

A utilização de uma metodologia com a definição de requisitos, modelagem e criação de artefatos e implementação do sistema são também objetivos do trabalho, pois são essenciais para garantir um bom projeto de *software*, que consequentemente possar ser utilizado, mantido, extendido e evoluído facilmente.

1.3 CONTRIBUIÇÕES

Este trabalho propõe o desenvolvimento de um pacote de *software* com uma preocupação na modelagem bem elaborada, incluindo requisitos e interações entre o usuário e o sistema (Diagrama de Casos de Uso), organização de dados e informações (Diagrama de Classes) e dinâmica de execução do sistema (Diagrama de Sequências). O trabalho de modelagem virá acompanhado de um protótipo que inclui algumas das funcionalidades dispostas nos requisitos.

O sistema beneficiará dois grupos de usuários: (i) desenvolvedores, com um modelo de sistema e uma documentação abrangente, permitindo sua manutenção, inclusão das funcionalidades levantadas nos requisitos e ainda não implementadas e extensão do modelo para novos requisitos; e (ii) pesquisadores, com um ambiente integrado que permitirá a utilização e execução de Análise Filogenética através de Redes Complexas por uma interface gráfica, armazenamento e organização de arquivos, salvamento e retomada de execuções das análises de forma transparente, direcionado para pesquisadores que não são da área de Ciência da Computação.

1.4 ESTRUTURA DO TRABALHO

Esta monografia está organizada da seguinte maneira:

No capítulo 2 apresenta-se a fundamentação teórica com seus principais conceitos e os passos para a execução do método de Análise Filogenética através de Redes Complexas: escolha das sequências proteicas, similaridade, construção e caracterização de redes, limiar crítico e entremeação, incluindo requisitos e a proposta.

No capítulo 3, mostra-se o projeto do sistema propriamente dito, e suas questões de escopo, organização das informações, casos de uso, modelagem de dados e sua dinâmica de execução através dos diagramas de sequência.

No capítulo 4 são mostrados detalhes de implementação, ambiente de execução, ferramentas de desenvolvimento, além de discutidos quais objetivos foram alcançados, e levantadas outras questões importantes e dificuldades encontradas.

Por fim, é apresentada a conclusão do trabalho e os trabalhos futuros (no capítulo 5), além de anexos e referências bibliográficas.

2 ANÁLISE FILOGENÉTICA ATRAVÉS DE REDES COMPLEXAS

Análise Filogenética é uma área da Biologia que tem crescido consideravelmente e despertado bastante interesse dos pesquisadores. Inicialmente usada para elucidar relações hierárquicas (Classificação e Taxonomia), a filogenética se expandiu, sendo usada hoje com inúmeros objetivos: criar novos grupos taxonômicos; reconstruir filogenias de organismos com representantes de diferentes áreas geográficas; buscar relações de ramificação com áreas de endemismo (Biogeografia histórica); investigar a evolução de espécies que interagem entre si, tais como hospedeiros e parasitas ou simbiontes, co-evolução de insetos e plantas; estudar evolução de caracteres em si mesmo, compreender melhor a dinâmica de populações, entre outros (SCHNEIDER, 2007).

A evolução do poder de processamento dos computadores permitiu que se pudesse comparar sequências proteicas ou nucleotídicas muito rapidamente, gerando um poder de análise abrangente no que tange as relações entre proteinas, enzimas, rotas metabólicas e consequentemente seres vivos. Essas análises, por conseguinte, geraram descobertas importantes que influenciaram o modo como enxergamos a evolução das espécies e a relação entre elas em geral.

Atualmente existem quatro métodos bem aceitos na literatura para Análise Filogenética: Análise Bayesiana, Análise de Parcimônia, Análise de Distâncias e Análise por Verossimilhança. Para uma compreensão sobre eles sugere-se a leitura de (DINIZ, 2010) e (SCHNEIDER, 2007).

A área de Física Estatística atualmente apresenta forte caráter interdisciplinar, permitindo a modelagem e análise de diversos sistemas que englobam outras áreas de conhecimento. A aplicação da Teoria das Redes Complexas em Bioinformática, apoiada pela Teoria dos Grafos, juntamente com conhecimentos de Física, Matemática e Biologia vem se mostrando como uma alternativa interessante na modelagem de sistemas biológicos, e serviu para que o FESC desenvolvesse um método para Análise Filogenética. A Figura 2.1 mostra o fluxograma geral sobre os passos da execução do método. As seções que seguem explicam estes passos com mais

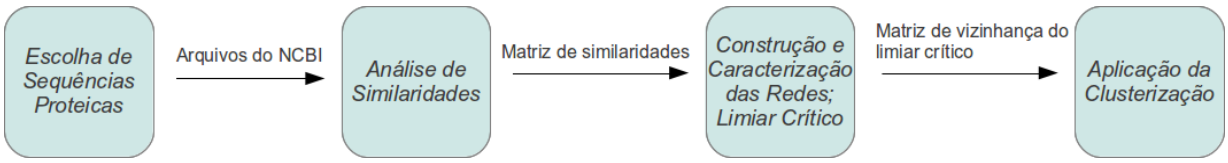

Figura 2.1: Fluxograma com os passos da execução do método de análise filogenética desenvolvido pelo FESC.

detalhes.

2.1 ESCOLHA DE SEQUÊNCIAS PROTEICAS

A primeira etapa do método consiste na escolha das sequências que serão utilizadas para a construção e caracterização da rede. Usualmente a escolha é feita a partir de um banco de dados biológico, disponível livremente na *web*. Durante o trabalho que necessitou da utilização do método, as sequências escolhidas vieram do NCBI (*National Center for Biotechnology Information*) (NATIONAL. . . , 2011), um banco de dados bastante utilizado no meio biológico, que contém dados de sequências nucleotídicas e sequências proteicas, entre várias outras informações. A escolha das sequências proteicas é feita utilizando como critério uma rota metabólica, separando as sequências pela enzima que as produzem na rota.

Esta é uma etapa manual, onde são feitas buscas por data e são realizadas determinadas filtragens no *site* do NCBI para obter as sequências desejadas, resultando em arquivos em um formato chamado GenBank. Um requisito importante para o sistema é a capacidade de receber um ou mais arquivos neste formato e a partir deles realizar uma filtragem e gerar a matriz de similaridades. O sistema deve levar em consideração que as sequências já foram escolhidas e as receber como entrada inicial. O resultado desta etapa devem ser arquivos filtrados e organizados para facilitar a identificação de organismos em etapas posteriores e proporcionar a execução da similaridade.

2.2 ANÁLISE DE SIMILARIDADES

Uma vez selecionadas as sequências, o próximo passo é executar a análise de similaridades entre as sequências proteicas e o armazenamento em um banco de dados relacional. Para isso é utilizada a ferramenta BLAST (*Basic Alignment Search Tool*) (ALTSCHUL et al., 1997) para comparação entre as sequências. A BLAST trabalha tanto com sequências nucleotídicas quanto proteicas. O interesse do uso da BLAST para o método em questão está na comparação de

sequências proteicas utilizando o alinhamento local, que compara duas sequências isoladamente e retorna uma porcentagem de similaridade entre elas.

A capacidade de operar com a BLAST é um requisito de grande importância neste contexto, tanto no envio das sequências para comparação quanto no recebimento e tratamento de seus resultados. A partir deles, pode-se construir uma matriz de similaridades (ANDRADE et al., 2011), onde as linhas e colunas representam as sequências e uma posição (i, j) da matriz armazena a similaridade entre elas. Desta matriz são construídas as redes que caracterizam as sequências, que serão descritas na próxima Seção.

2.3 CONSTRUÇÃO E CARACTERIZAÇÃO DAS REDES

O conceito de rede está associado ao de um grafo, que é um par $G = (V,E)$ de conjuntos onde $E \subseteq [V]^2$, então os elementos de E são subconjuntos de dois elementos de V. Para evitar ambiguidades de notação, devemos assumir que $V \cap E = \emptyset$. Os elementos de V são os vértices (ou nós ou pontos) e os elementos de E são suas arestas (ou linhas) (DIESTEL, 2010).

Podemos dizer que o estudo de uma rede é o estudo de um grafo, mas em uma escala maior. Redes podem ser tidas como grafos muito grandes e complexos, onde se torna necessário o uso de técnicas estatísticas para a sua análise (BESSA; SANTOS; COSTA, 2008). Na literatura de Redes Complexas, em geral, precisamos determinar qual rede de um dado conjunto melhor o caracteriza, seguindo uma propriedade. Essa escolha se dá pela relação entre ruído e informação (BARABASI; OLTVAI, 2004).

Partindo da matriz de similaridades, podemos construir até 101 matrizes de adjacência, relacionando cada matriz a uma escala de similaridade de 0% a 100%, se levarmos em consideração que podemos escolher um valor limiar e construir a rede baseada nele. Por exemplo, se escolhermos o valor 60 como limiar, construiremos uma rede onde haverá uma aresta entre dois vértices (sequências) se e somente se a similaridade entre eles for maior ou igual a 60. Uma rede com limiar muito baixo apresenta muito ruído, ou seja, muitas arestas, o que leva a pouca possibilidade de identificação de suas características. Por outro lado, uma rede com limiar muito alto apresenta pouca informação, ou seja, poucas arestas, levando também a pouca possibilidade de identificação de suas características.

Partindo das matrizes de adjacência podemos também criar matrizes de vizinhança (ANDRADE; PINHO; LOBÃO, 2009), que podem ser obtidas calculando-se os caminhos mínimos (BESSA; SANTOS; COSTA, 2008) entre os vértices da rede ou através da multiplicação booleana de matrizes de adjacência de ordens consecutivas (ANDRADE; MIRANDA; LOBÃO,

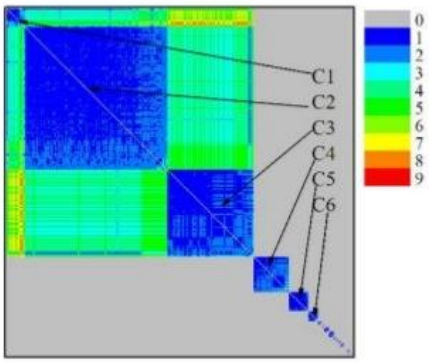

Figura 2.2: Exemplo de Matriz de Vizinhança no formato de Matriz de Cores.

2006). Na Figura 2.2, temos um exemplo de matriz de vizinhança, onde cada cor representa um valor de distância entre dois vértices. Pela Figura, temos que se a distância entre dois vértices é 1, o ponto que representa a relação entre eles recebe a cor azul escura. Se for 7, por exemplo, receberá a cor amarela.

Alguns programas utilizados geram as matrizes em memória e as excluem após determinados cálculos, e outros simplesmente as salvam em certa estrutura de diretórios. A possibilidade de dar ao usuário o poder de gerenciar essas matrizes e realizar operações diversas, como compará-las ou visualizar gráficos a partir delas ao longo de um procedimento de análise é algo importante que deve estar incluído em um ambiente integrado de análise filogenética.

2.4 DETERMINAÇÃO DO LIMIAR CRÍTICO

Já se sabe que o limiar crítico de uma rede revela a melhor relação entre ruído e informação. Existem diversas maneiras de determiná-lo. Uma delas é o método das distâncias (ANDRADE; PINHO; LOBÃO, 2009). Nele, calcula-se a distância euclidiana entre matrizes de vizinhança de limiares consecutivos (ANDRADE et al., 2011). A matriz que apresentar a maior distância entre a matriz de limiar consecutivo ao seu será a escolhida como a matriz do limiar crítico, para representar a rede. A literatura relacionada a Redes Complexas mostra que é necessário escolher a melhor rede que representa o conjunto, de acordo com uma propriedade. No método em questão, é a rede que revela maior possibilidade de análise de seu caráter modular. A rede do limiar crítico cumpre bem este papel.

Na Figura 2.3, temos um exemplo de resultado do cálculo das distâncias entre matrizes de vizinhança, onde o limiar crítico é representado pelo pico do gráfico, ou seja, o valor 51. A rede que representa todo o conjunto é então a rede limiar 51, que será utilizada na última etapa do

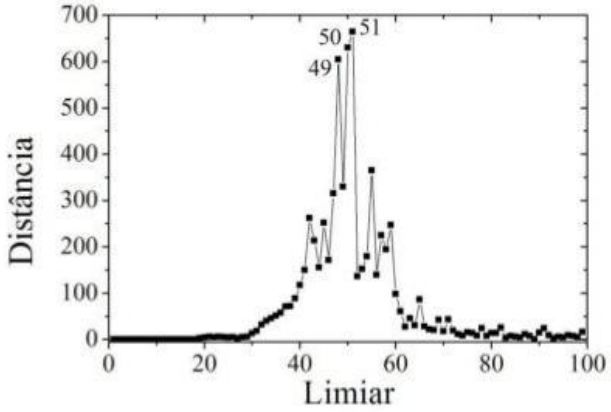

Figura 2.3: Resultado da execução do método das distâncias entre limiares consecutivos, resultando em um gráfico Distância x Limiar.

processo: a clusterização.

O método das distâncias pode ser utilizado para a determinação do limiar crítico, mas isto pode ser feito através de outros métodos, como a verificação da variação do tamanho do maior *cluster* da rede em função da variação do limiar. Portanto, um ambiente integrado deve fornecer a possibilidade de escolha do método de análise dos limiares.

2.5 APLICAÇÃO DA CLUSTERIZAÇÃO

A matriz de vizinhança da rede do limiar crítico é a entrada necessária para a realização da clusterização, que é feita através do método de detecção de comunidades de Newman e Girvan (NEWMAN; GIRVAN, 2004) para identificar a estrutura modular da rede. O método consiste em determinar a aresta mais importante da rede – aquela por onde passa a maior quantidade de caminhos mínimos de cada par de vértices por toda a rede – e eliminá-la, repetindo a operação até que não existam mais arestas na rede. Estas operações consistem no conceito de *Betweenness* ou Entremeação (ANDRADE et al., 2008).

Conforme esta operação é feita, é possível construir um histograma de remoção de arestas ou dendrograma, que começa com uma linha representando toda a rede e à medida em que as arestas são removidas a rede vai se repartindo em módulos e a linha inicial vai sofrendo bifurcações. Observando a Figura 2.4, por exemplo, entre o vértice 100 e 120 há uma bifurcação, indicando a divisão de um módulo em dois.

O dendrograma auxilia na identificação de comunidades (módulos) e propicia as análises biológicas dos resultados, além da comparação com outros métodos da Biologia, visto que seu

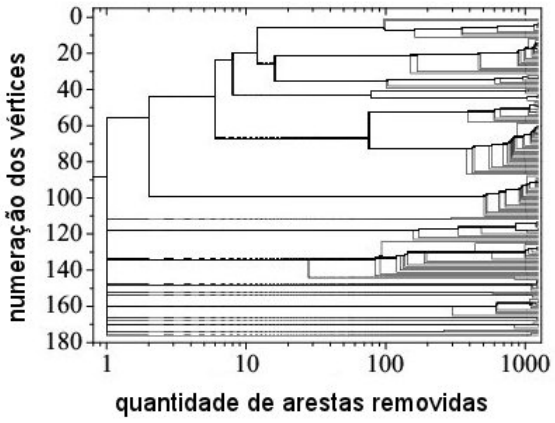

Figura 2.4: Exemplo de dendrograma para o limiar crítico 51, onde temos a numeração dos vértices pela quantidade de arestas removidas, em escala logaritmica.

formato de árvore em que as linhas mais à direita representam cada uma das sequências proteicas (vértices da rede) é também o formato de saída dos outros métodos de Análise Filogenética.

Para a geração do dendrograma é necessária a escolha do limiar pelo usuário, e seu algoritmo recebe como entrada a matriz de vizinhança referente a este limiar. A execução desta etapa é a mais demorada, variando de cinco minutos a mais de três meses, a depender do tamanho da rede.

O método completo em questão é explicado com mais detalhes em (GÓES-NETO et al., 2010) e (ANDRADE et al., 2011). Ele utiliza uma série de programas, cada um seguindo padrões de entrada e saída e arquivos de conFiguração particulares. Tais características dificultam sua utilização por pesquisadores de outras áreas que não Física ou Ciência da Computação, além de deixar a seu cargo a tarefa de localizar, distribuir e organizar os arquivos. Para otimizar o processo, seria mais interessante que o seu gerenciamento fosse realizado por um ambiente de fácil utilização e que mantivesse disponíveis as informações por meio de uma camada de persistência, deixando a questão da organização dos dados transparente para o pesquisador.

3 *MODELAGEM DO AMBIENTE INTEGRADO DE APOIO À ANÁLISE FILOGENÉTICA ATRAVÉS DE REDES COMPLEXAS*

Da forma como está estruturado atualmente, o processo de análise filogenética através de redes complexas pode ser resumido como uma série de execuções de programas independentes entre si que recebem arquivos de entrada em determinados formatos, realizam seu processamento e geram outros arquivos de saída. O desenvolvimento de um ambiente integrado que busque facilitar o controle de execução do processo torna-se importante nesse contexto. Nas seções a seguir são detalhadas a metodologia de desenvolvimento adotada e os artefatos de modelagem gerados.

3.1 METODOLOGIA DE DESENVOLVIMENTO

O desenvolvimento interativo e incremental tem se mostrado como um processo de desenvolvimento alternativo ao modelo em cascata, e traz grandes vantagens por garantir ciclos menores de desenvolvimento e uma divisão do problema em problemas menores (PRESSMAN, 2005). A metodologia de desenvolvimento adotada foi o modelo de desenvolvimento em espiral (BOEHM, 1986). A espiral combina ideias do desenvolvimento interativo com alguns aspectos do modelo em cascata.

O modelo em espiral é baseado em refinamento contínio do produto a partir de definição de requisitos e análises, projeto de *software*, e implementação. A cada interação dentro do ciclo, o produto gerado é considerado uma extensão de um produto mais antigo. Esse modelo utiliza muitas das fases do modelo em cascata, na mesma ordem, separados por planejamento, análise de riscos, construição de protótipos, testes e validação, como é mostrado na Figura 3.1

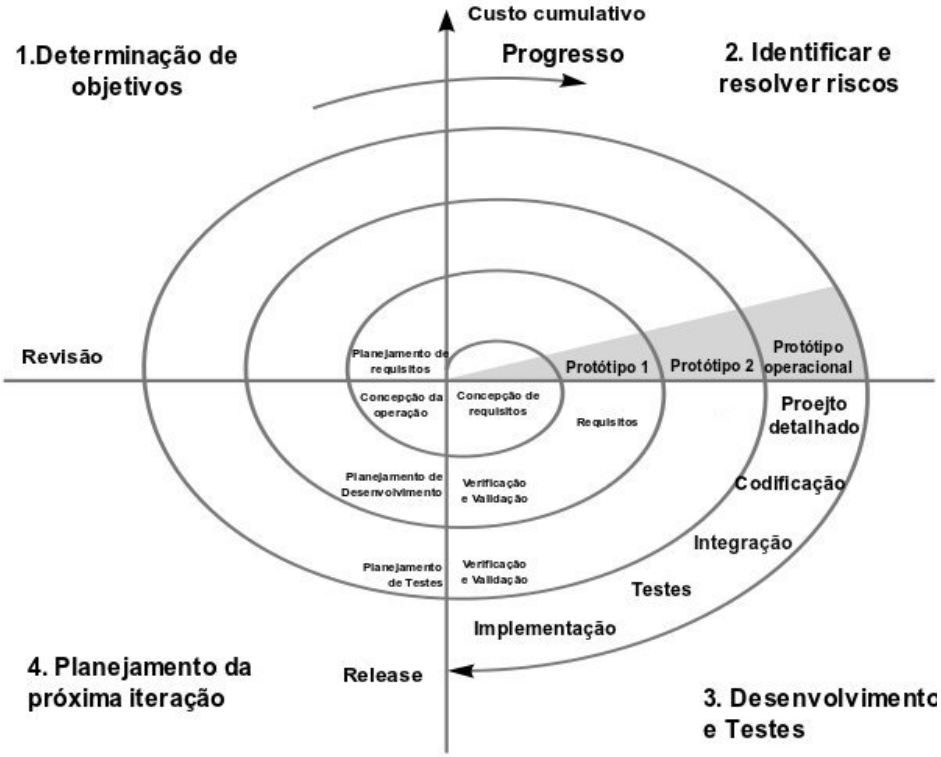

Figura 3.1: Modelo de desenvolvimento em espiral (BOEHM, 1986).

3.2 LEVANTAMENTO DE REQUISITOS

A partir do que foi descrito no Capítulo 2, os requisitos mais importantes são a eliminação da necessidade de organização de arquivos intermediários pelos usuários e a conversão de formatos entre as execuções. Para agilizar a execução, é importante também a geração de gráficos em determinadas etapas. A tabela 3.1 mostra de forma consolidada os requisitos funcionais levantados e que devem ser satisfeitos no processo de modelagem e construção de um AIAF - Ambiente Integrado de apoio à Análise Filogenética Através de Redes Complexas.

A partir dos requisitos identificados na Tabela 3.1, podemos propor um ambiente integrado para gerenciar o método de Análise Filogenética por Redes Complexas descrito anteriormente. Para garantir o controle das etapas do processo e ao mesmo tempo evitar problemas com padrões e arquivos de configuração, o sistema precisa seguir três importantes requisitos não-funcionais:

- **Usabilidade:** é um fator importante por tornar o método passível de utilização por pessoas com pouco ou quase nenhum conhecimento de programação neste caso específico, e para gerar uma boa experiência do usuário e fazer com que seu uso seja agradável no caso geral. Segundo (NIELSEN, 1993), interfaces de usuário são uma parte muito mais importante para computadores em geral do que já foram um dia, e hoje respondem por mais

Tabela 3.1: Tabela de requisitos funcionais do AIAF

Requisito	Descrição
Filtragem de arquivos no formato GenBank	O sistema deve receber um diretório e ler dele arquivos no formato do GenBank, e a partir deles criar os organismos, as sequências com nome, identificador, código e outras informações
Controle da análise	O sistema deve prover ao usuário a possibilidade de criar, salvar ou carregar uma análise
Salvamento de organismos e sequências	O sistema deve salvar os organismos e as sequências para posterior consulta. Deve ser criada uma estrutura de armazenamento que livre o usuário da necessidade de instalar sistemas de bancos de dados relacionais
Escolha de sequências	O sistema deve permitir a escolha de sequências para a geração da matriz de similaridades
Geração de matrizes de adjacência	O sistema deve gerar matrizes de adjacência com base no limiar fornecido pelo usuário, também mostrando o gráfico de uma forma opcional
Geração de matrizes de vizinhança	O sistema deve gerar matrizes de vizinhança com base no limiar fornecido pelo usuário, também mostrando o gráfico de uma forma opcional
Análise de limiares	O sistema deve fornecer opções de métodos para a análise de limiares, mostrando obrigatoriamente o gráfico como resultado
Execução da clusterização	O sistema deve fornecer opções de métodos de clusterização, que depende da escolha de um limiar pelo usuário, além de mostrar o gráfico de seu resultado quando requisitado
Visualização do gráfico completo da rede	O sistema deve permitir a visualização do gráfico completo da rede, após a escolha das comunidades, recuperando as informações salvas durante o processo de salvamento das sequências e organismos
Congruência	O sistema deve permitir a comparação do resultado do método com o resultado de outros métodos de análise filogenética, além da comparação com o resultado pelo próprio método com outro conjunto de dados
Exportação de dados	O sistema deve permitir a exportação de dados necessários à geração de gráficos para o uso pelos usuários de ferramentas profissionais de plotagem

de 45% do código-fonte de um *software*. Podemos também enquadrar aqui as interfaces de programação, levando a uma fácil manutenção e possibilitando extensibilidade.

- **Gerenciamento da Informação:** torna possível a criação, importação e exportação de dados contendo as execuções necessárias para as análises, onde os dados podem ser agrupados em um formato de arquivo possível de ser transportado entre múltiplas máquinas e contendo todas as informações necessárias para a retomada de execuções.

- **Transparência:** não menos importante, permite que as pessoas que estão executando o método não precisem se preocupar com detalhes de mais baixo nível de execução, como os locais em que os arquivos estão dispostos e situações relacionadas à conversão de formatos e adaptações diversas de dados.

No levantamento de requisitos, foram determinados os objetivos e alternativas e definido o escopo do sistema baseado nos objetivos do projeto, dispostos na Seção 1.2. Em seguida foi realizada a modelagem, com os diagramas de casos de uso, classes e sequência. Também foi definida a forma como o sistema organizaria e realizaria a persistência de seus dados, levando em consideração os requisitos definidos. A última etapa do primeiro ciclo da espiral consistiu no planejamento do desenvolvimento, implementação do protótipo e testes.

Este Capítulo trata das etapas de modelagem, que foram realizadas utilizando a linguagem UML (UML, 2011). A implementação do protótipo e os testes são tratados no Capítulo 4.

3.3 CASOS DE USO

Esta seção discute os casos de uso do AIAF, que podem ser vistos na Figura 3.2.

O processo de análise atual utiliza um banco de dados relacional para armazenamento de informações sobre as sequências proteicas. Isto força o usuário a instalar um banco de dados e configurá-lo corretamente. Um dos requisitos importantes, como visto no Capítulo anterior, é a retirada desta necessidade e, por consequência, a criação de uma nova forma de organização e disposição dos dados.

Diversos ambientes de *software* trabalham com o conceito de projeto. Um projeto representa um conjunto de operações feitas durante a execução do sistema, podendo-se salvar e recuperar o andamento das atividades. Tal fato é possível a partir de uma estruturação de dados para um correto procedimento de salvamento e posterior recuperação das informações. Trabalhar com projeto se torna importante na medida em que se pode dar ao usuário a possibilidade

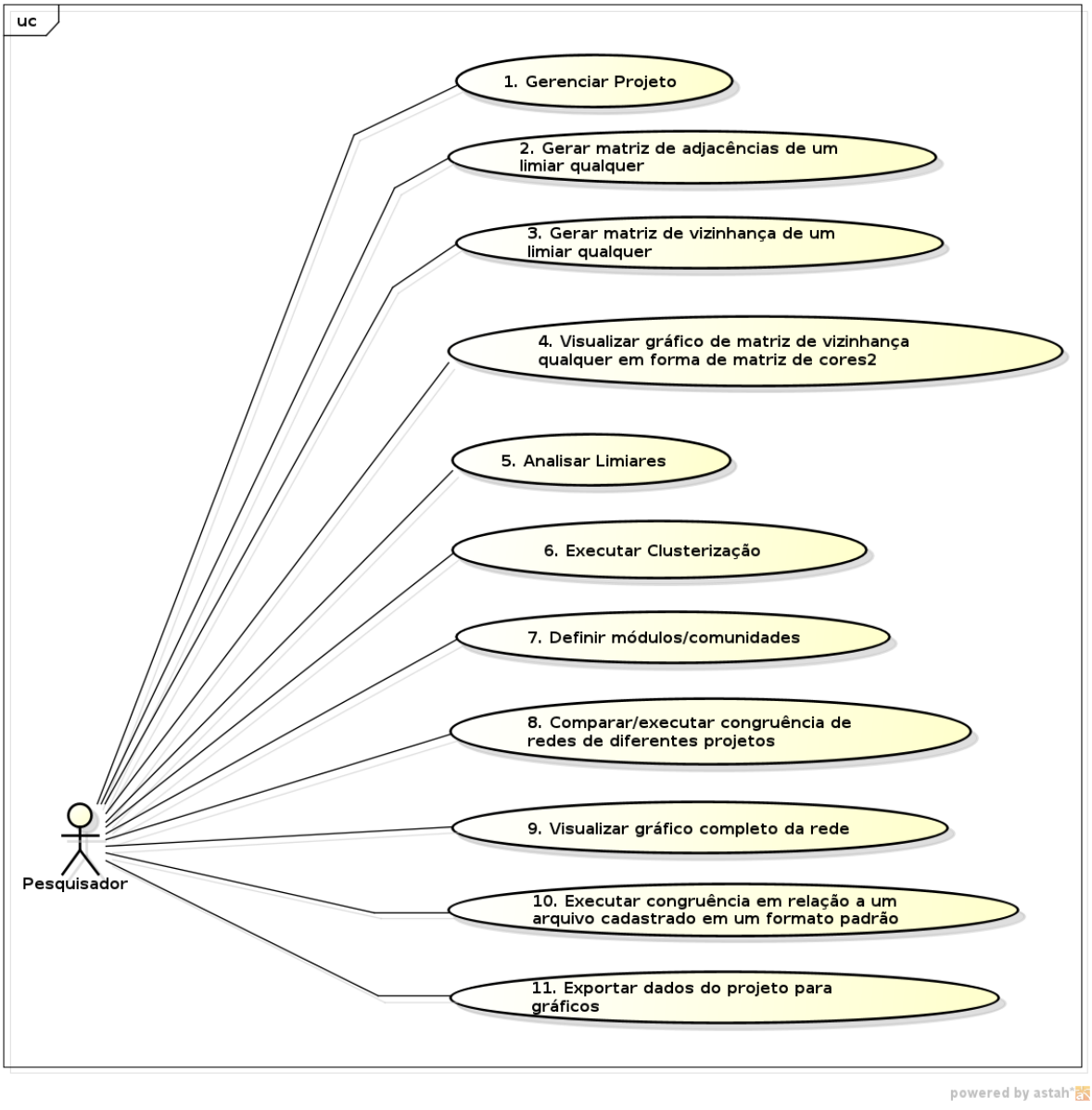

Figura 3.2: Diagrama de casos de uso do AIAF.

de criar diversas análises, importar seus resultados e armazenar suas informações em qualquer etapa do processo. Isto permite uma interoperabilidade entre máquinas importante, e a organização de um arquivo que possa conter os dados necessários para o carregamento de um projeto salvo torna-se essencial. O sistema então permitirá aos seus usuários a criação de novos projetos e o carregamento de projetos existentes (caso de uso 1).

A comunicação com módulos externos deve ser transparente e a organização de classes deve fornecer alternativas viáveis para a inclusão de novos tipos de análises, como por exemplo, a escolha do limiar crítico pode ser feita por uma análise de distâncias, como também por uma análise da variação do tamanho do maior *cluster* da rede, ou seja, o sistema deve suportar extensibilidade e a fácil troca de módulos externos, mantendo os padrões de comunicação.

A partir de certo momento do processo, passa-se a trabalhar apenas com matrizes. Há a perda de informações com relação aos organismos e sequências em que se está trabalhando. É necessário, então, que o sistema guarde estas informações e possa exibir para os usuários gráficos mais ricos, contendo informações sobre sequências e espécies com que se está trabalhando. Por exemplo, na exibição da rede (grafo), o sistema poderia exibir, ao se passar o *mouse* sobre um vértice, informações sobre a sequência e o organismo a que o vértice está representando (caso de uso 9). O usuário poderá também gerar matrizes de adjacência e vizinhança a qualquer momento (casos de uso 2 e 3), além de ver o gráfico de qualquer matriz de vizinhança em formato de matriz de cores (caso de uso 4).

Em várias etapas da execução do processo de análise atual, é necessário mudar de ambiente (tipicamente, de Linux para Windows) para efetuar certo processamento, como por exemplo visualizar gráficos em programas específicos (ORIGIN, 2011), que são necessários para certas tomadas de decisões que influenciam em quais dados serão escolhidos para a execução das próximas etapas do mesmo. Isto é prejudicial pois gera uma troca de contexto. O sistema, então, pode gerar esses gráficos (matrizes de vizinhança em forma de matrizes de cores, matrizes de adjacências em forma de grafos, gráficos de resultados de análises como distâncias, gráfico do dengrograma), mesmo que com qualidade um pouco inferior, para que a continuação do processo se dê de forma mais rápida. Esta funcionalidade representa uma preocupação que deve estar presente no momento da modelagem para tornar o processo mais ágil e transparente para o usuário. Ao mesmo tempo, o sistema pode fornecer como opção de exportação os arquivos que são utilizados por programas profissionais de plotagem, essenciais para a utilização em artigos científicos e afins (caso de uso 11).

A análise de limiares (caso de uso 5) resulta na geração de um gráfico que será utilizado como base para a definição do limiar crítico e o prosseguimento do processo, com a

clusterização (caso de uso 6) e a definição de comunidades (caso de uso 7).

Outro detalhe importante é a possibilidade de comparação entre o método atual desenvolvido pelo FESC e outros métodos já consolidados de análise filogenética, listados no Capítulo 2, além de poder comparar com outro projeto criado pelo próprio sistema (casos de uso 8 e 10). Para prosseguir com a modelagem, se faz necessária a organização das informações através de um diagrama de classes, mostrado na próxima Seção.

3.4 MODELAGEM DOS DADOS

A organização das informações do sistema é um passo importante para que se tenha qualidade e se atenda aos requisitos propostos. Torna-se de grande importância a criação de um diagrama de classes para definir classes e suas responsabilidades, além de seus aspectos funcionais (métodos) e os dados (atributos) em que cada uma é responsável.

Após análise das informações manipuladas no ambiente de análise atual e considerando os casos de uso definidos na Seção anterior, foi criado um diagrama de classes, cuja versão simplificada pode ser vista na Figura 3.3. O diagrama de classes completo pode ser encontrado no Apêndice A.

Na Figura 3.3 as classes estão agrupadas para facilitar o entendimento. A numeração adotada foi intencional pelo mesmo motivo. As três classes mais importantes estão no grupo 1. Elas são responsáveis por garantir usabilidade, gerenciamento de informação e transparência. No grupo 2 estão as classes que tratarão da parte inicial do método, onde há a filtragem dos dados provenientes do GenBank e o salvamento de sequências e organismos, além da execução da ferramenta BLAST para a comparação de sequências e geração da matriz de similaridades. Os grupos 3 e 4 demonstram de forma clara o poder de extensibilidade do sistema, onde de pode incluir novas matrizes ou novas análises de limiares ou de clusterização, por exemplo, através da simples criação de uma classe e utilização de herança. *BLAST* foi definido como pacote por se tratar de uma ferramenta externa. *GUI* (*Graphical User Interface*) e *Persistence* foram definidos da mesma maneira pois seu detalhamento é irrelevante no contexto da análise da modelagem de classes, mas fazem parte do núcleo do AIAF. A seguir teremos um detalhamento melhor sobre as classes.

A classe principal do AIAF é a classe *Project*. Todas as interações do usuário com a *GUI* passam por ela, que controlará operações com matrizes, análises, tratamento das informações biológicas e dos dados de entrada. Através dela é possível criar novo projeto, abrir projeto existente e salvar projeto corrente (caso de uso 1). Um projeto para o sistema reúne todos os

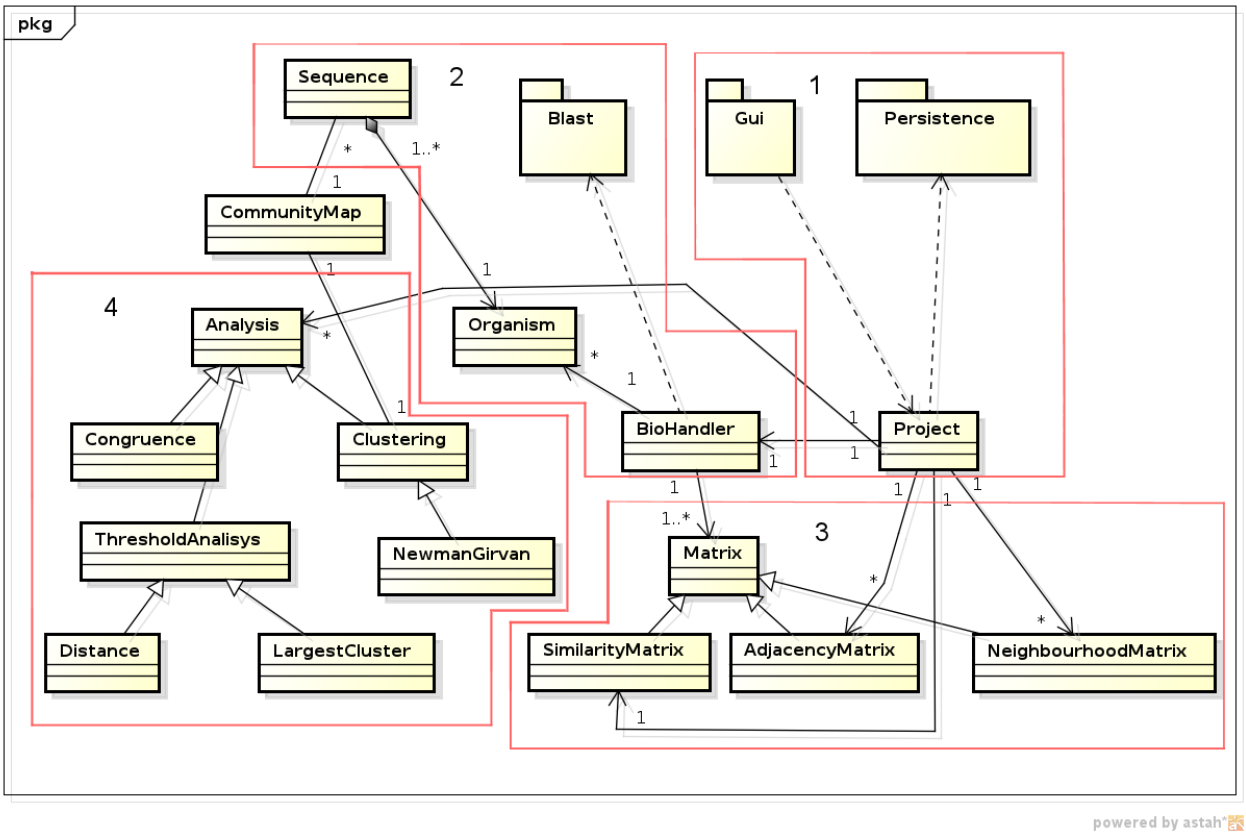

Figura 3.3: Diagrama de classes simplificado do AIAF.

dados de informações de execução de um conjunto de dados, desde a escolha de sequências, geração de matriz de similaridade, matrizes de adjacência e vizinhança, gráficos, definição de comunidades, clusterização, congruência. A classe *Project* também lida com o pacote *Persistence*, que mapeia as classes do sistema em uma camada de persistência, via armazenamento em disco. Sua função inclui serializar e desserializar os objetos do sistema, além de prover a organização dos dados do projeto, atuando de forma transparente ao usuário. Os dados de vários objetos são salvos, embora todo o controle para persisti-los tenha que passar pela classe *Project*. Como resultado, o usuário obtém um arquivo que representa o mapeamento feito pela camada de persistência em um sistema de arquivos comprimido e de extensão própria (.aiaf).

A filtragem do banco de dados biológico (sequências proteicas que estão em vários arquivos no diretório especificado pelo usuário) é realizada pela classe *BioHandler*, que a partir desta operação instancia objetos da classe *Organism* e da classe *Sequence*, ou seja, cria organismo e associa sequências a eles. Neste contexto, um organismo pode conter uma ou mais sequências, o que explica a composição realizada no diagrama de classes. Ela utiliza o pacote *Blast*, que realiza as comparações entre sequências, para gerar e instanciar a matriz de similaridades.

O AIAF possui três matrizes, representadas pelas classes *SimilarityMatrix*, *AdjacencyMa-*

trix e *NeighbourhoodMatrix*. As três herdam da classe *Matrix* por conterem características em comum. Só é possível ter uma matriz de similaridades por projeto, pois como um projeto está associado a execuções de determinado conjunto de sequências escolhido pelo usuário, há apenas uma matriz de similaridades possível para qualquer conjunto de sequências. Em relação às demais (adjacência e vizinhança), é possível haver mais de uma. Outros tipo de matrizes poderão ser facilmente incorporadas ao sistema através da adição de uma nova classe que será uma especialização da classe *Matrix*, o que explica o fato de a classe *Matrix* ser um classe abstrata. É possível gerar matrizes de adjacência ou de vizinhança, além da visualização de seus respectivos gráficos a qualquer momento após a escolha das sequências e geração da matriz de similaridades, de acordo com os casos de uso 2, 3 e 4. A classe *Project* contém métodos para a geração das matrizes, o que pode ser visto no diagrama de classes completo no Apêndice A. Após a geração das matrizes, a camada de persistência as salva em disco de forma transparente.

A classe *Analysis* é uma classe abstrata que concentra todos os tipos de análises que podem ser feitas sobre as matrizes relacionadas às redes proteicas. Existem três tipos de análises: Análise de Limiares (classe *ThresholdAnalysis*), Clusterização (classe *Clustering*) e Congruência (class *Congruence*). Como descrito anteriormente, existem duas formas atualmente de analisar limiares: Distâncias (classe *Distance*) e tamanho do maior cluster (classe *LargestCluster*). Elas são subclasses de *ThresholdAnalysis*, que também é abstrata. Essa abordagem permite que outros métodos de análise de limiares possam ser adicionados de forma facilitada por meio de herança. O mesmo acontece com a clusterização: a classe *Clustering* é abstrata, e tem *Newman-Girvan* como classe filha, que implementa o único método de clusterização usado no processo até o momento (casos de uso 5 e 6).

A análise de congruência permite a comparação da divisão de comunidades entre diversos projetos executados sob o AIAF, como também entre outros métodos de análise filogenética, através do fornecimento de um arquivo em formato padrão. Isso é possível graças à classe *Congruence*, que é uma especialização da classe *Analysis* (casos de uso 8 e 10).

Outras funcionalidades do sistema incluem a visualização do gráfico completo da rede, que utiliza a classe *CommunityMap* para desenhar os vértices que pertencem a uma mesma comunidade de determinada cor (caso de uso 8); e a possibilidade de exportar dados do projeto para gráficos em um formato de programas plotadores, utilizando-se de atributos de diversas classes, a depender do gráfico que se deseja (caso de uso 11).

O diagrama de classes permite ter uma visão estática das informações do sistema, enfatizando os seus relacionamentos e os métodos que as irão manipular. Na próxima Seção serão apresentados e discutidos os principais diagramas de sequência, que fornecem uma visão

dinâmica da interação entre as diversas classes do modelo ao longo do tempo.

3.5 DIAGRAMAS DE SEQUÊNCIA

Como discutido na Seção anterior, a classe *Project* é a classe principal do sistema, para onde os eventos captados pela interface gráfica são enviados. Nesta Seção serão abordados três importantes diagramas de sequência do AIAF: criar novo projeto, gerar matriz de adjacência de um limiar qualquer e analisar limiares. Os casos de uso 1, 4, 6, 7, 8, 9, 10 e 11 são autoexplicativos, e o caso de uso 3 é análogo ao caso de uso 2. Será apresentada uma discussão sobre sua dinâmica. Outros diagramas de sequência podem ser encontrados no Apêndice B.

O primeiro diagrama de sequência, visto na Figura 3.4, se refere à criação de um novo projeto (caso de uso 1). Primeiro, qualquer projeto que esteja em execução é finalizado e o pacote *Persistence* salva qualquer alteração nos objetos do sistema. Uma decisão de projeto importante é que após a instanciação de uma nova classe *Project* é criado também um novo objeto *BioHandler*, que já filtra os dados passados pelo usuário, criando objetos *Organism* e *Sequence*. Após isto, *Persistence* salva os dados do projeto.

Em seguida o usuário escolherá as sequências com que a matriz de similaridades será gerada, onde *BioHandler* passará as sequências, duas a duas, para que o pacote *Blast* possa calcular as porcentagens, e então o primeiro possa montar a matriz de similaridades. Por fim, os dados do projeto são salvos em disco pela classe de persistência.

Para gerar uma matriz de adjacências (caso de uso 2) o usuário entra com um valor limiar (ou uma lista de limiares para o caso da geração de múltiplas matrizes). Haverá então um *loop* na classe *Project* que instanciará objetos da classe *AdjacencyMatrix*, e então usará *Persistence* para salvar novos dados em disco, conforme a Figura 3.5.

Para analisar limiares (caso de uso 5), o usuário precisa informar qual análise quer executar. O método do FESC de análise filogenética dispõe atualmente de duas formas para a realização desta análise, mas existem vários outros e que podem ser adicionados ao AIAF. Atualmente o usuário deve optar, então, pelo método de distâncias ou tamanho do maior cluster. A depender da escolha será instanciado pela classe *Project* uma classe *Distance* ou *LargestCluster*. A análise é realizada passando-se a matriz de similaridades e, ao seu final, é invocada uma biblioteca gráfica para a exibição do gráfico para o usuário como resultado da análise. A Figura 3.6 mostra seu diagrama de sequência.

O próximo Capítulo mostrará os resultados deste trabalho, seu produto gerado, as ferra-

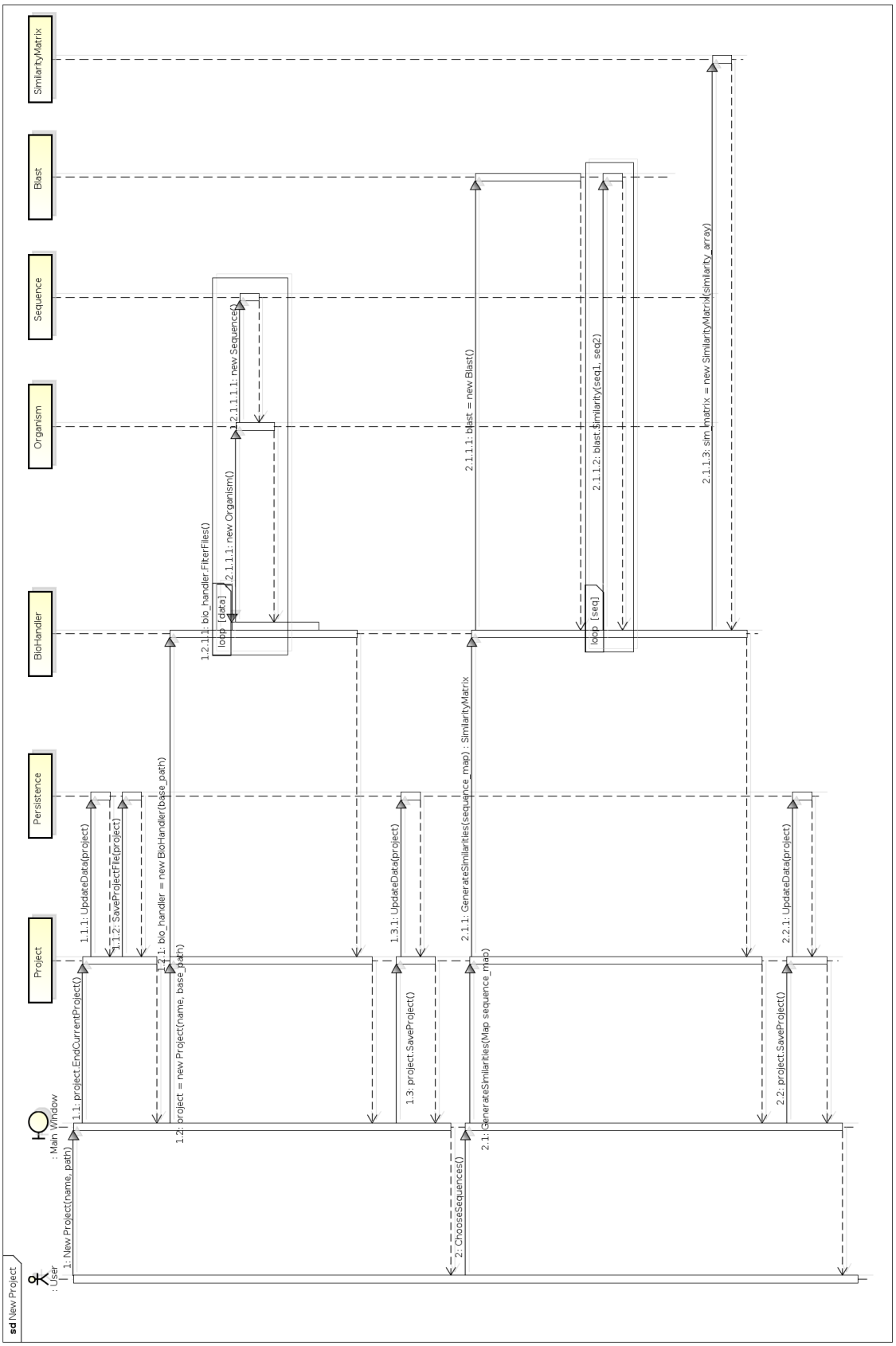

Figura 3.4: Diagrama de sequências para a parte de criação de um projeto referente ao caso de uso 1: gerenciar projeto.

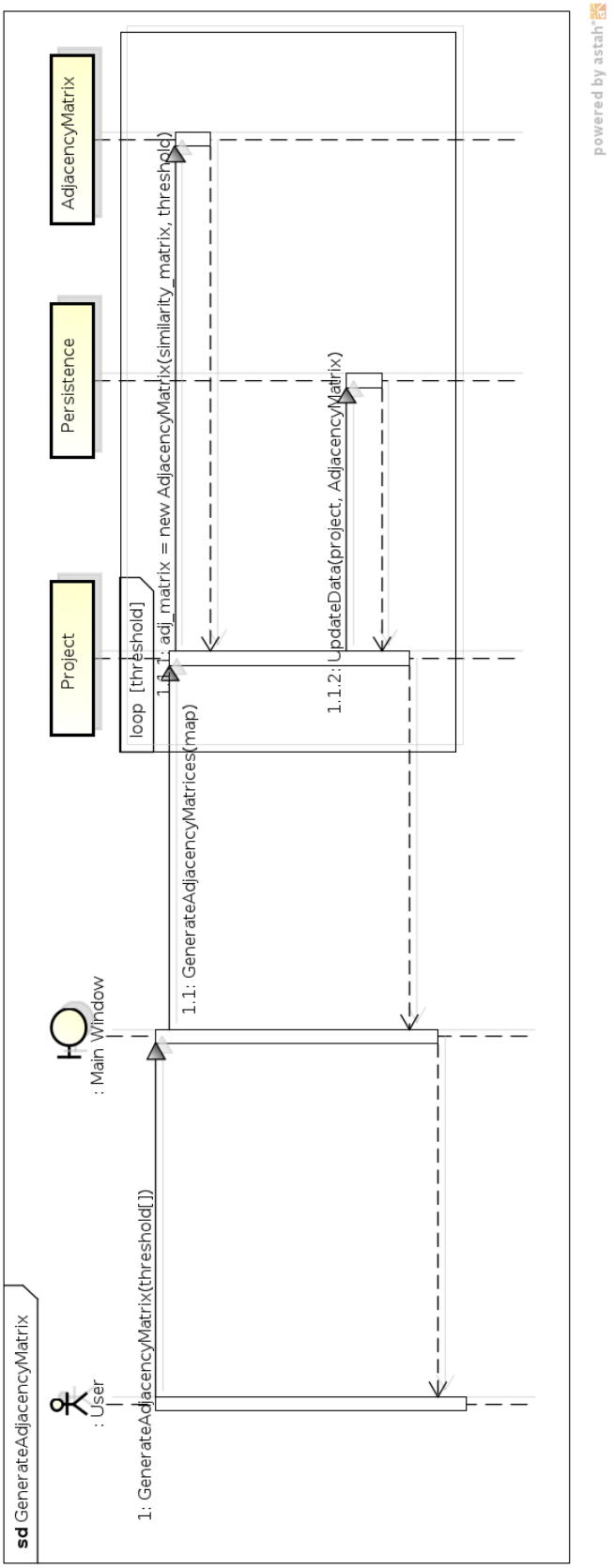

Figura 3.5: Diagrama de sequências para o caso de uso 2: gerar matriz de adjacências de um limiar qualquer.

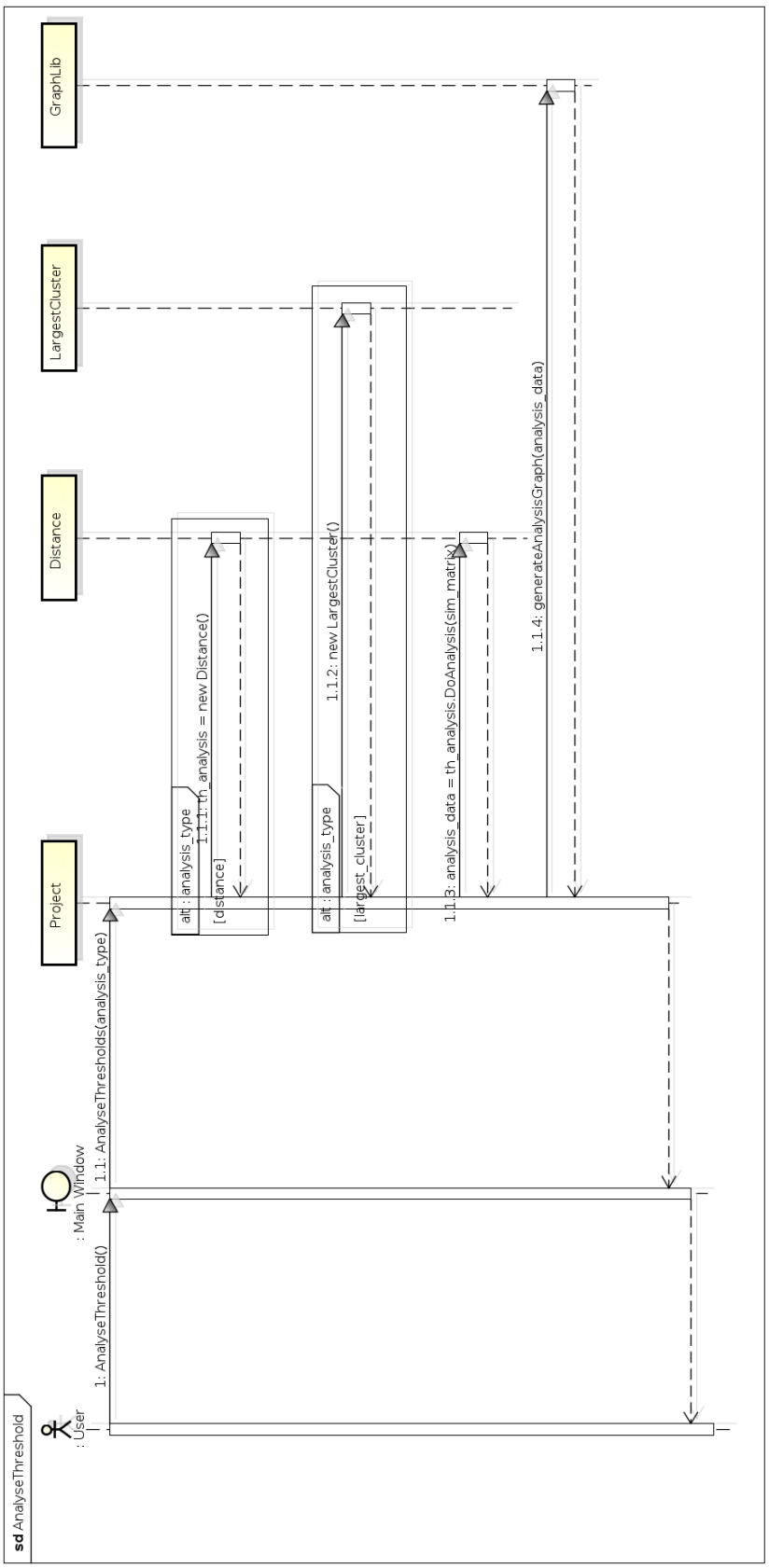

Figura 3.6: Diagrama de sequências para o caso de uso 5: analisar limiares.

mentas e o ambiente de execução utilizados, características de implementação e discutirá os resultados em relação aos objetivos propostos pelo trabalho.

4 IMPLEMENTAÇÃO DO PROTÓTIPO

Neste Capítulo são apresentadas de forma detalhada as técnicas utilizadas para o desenvolvimento do AIAF.

4.1 AMBIENTE DE DESENVOLVIMENTO

O ambiente operacional original envolve uma série de ferramentas e linguagens: exige um ambiente GNU/Linux, um compilador Fortran, um compilador C, pacotes para a execução de *scripts* Perl e a biblioteca *BioPerl*, além de um banco de dados relacional, neste caso o MySQL. Após o *download* das sequências a partir do banco de dados GenBank, as mesmas eram colocadas em um mesmo diretório, onde uma série de *scripts* Perl eram executados, realizando os passos necessários para o armazenamento das sequências no banco de dados e a posterior geração da matriz de similaridades.

De posse da matriz de similaridades, é necessária a execução de outro programa para a geração das redes de diversos limiares e análise dos mesmos para que fosse possível a escolha de um limiar crítico e então a continuação do método. Utilizando então a matriz de vizinhança da rede do limiar crítico, a clusterização é aplicada, gerando como resultado um dendrograma (histograma de remoção de arestas, indicando modularização da rede). O dendrograma, em conjunto com a matriz de cores e as sequências permitem análises biológicas sobre os resultados.

Os módulos permanecem os mesmos, mas o controle do processo é todo gerenciado pela GUI, a classe *Project* e o pacote *Persistence*, com auxílio de outras classes na execução e comunicação dos módulos externos. A Seção 4.2 detalha como se dá este gerenciamento.

A área de Bioinformática hoje em dia tem utilizado bastante Python (PYTHON..., 2011) no lugar de Perl para comparações entre sequências, utilização de algoritmos genéticos, entre outras técnicas na pesquisa em geral. É uma linguagem bastante rica e poderosa, com uma grande facilidade de uso, compreensão e desenvolvimento, rapidez na implementação e na pro-

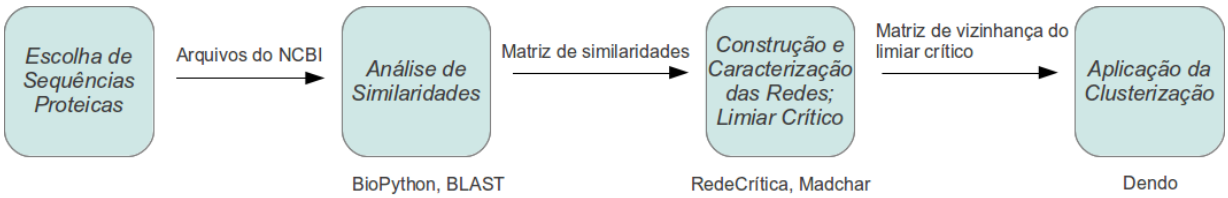

Figura 4.1: Ferramentas utilizadas em cada etapa do processo.

totipagem, bibliotecas robustas para o desenvolvimento de *software* científico, cálculos diversos e geração de gráficos, além de ser considerada uma "linguagem cola" por se comunicar facilmente com outras linguagens.

Para a modelagem foi utilizado o Astah Community (ASTAH..., 2011). Para a criação das janelas foram testados o Tkinter (PYTHON..., 2011), o Gtk (THE..., 2011) e o Qt (QT..., 2011). O Qt foi escolhido pela sua quantidade de recursos, facilidade de utilização, desenvolvimento e manutenção, total suporte à linguagem Python através do PyQt (PYQT..., 2011), além de fornecer o *Designer*, um bom ambiente de autoria para a criação de janelas e interface gráfica em geral. A utilização de Qt também permite a possibilidade de execução do AIAF em ambientes Windows e Mac, faltando apenas a adaptação do *back-end* para um suporte completo a essas outras plataformas.

A utilização da linguagem Python para o desenvolvimento do AIAF acarretou certas consequências. Uma delas foi a substituição do *BioPerl* pelo *BioPython* (BIOPYTHON, 2011), um conjunto de ferramentas para computação biológica feito em Python, equivalente ao *BioPerl* para *Perl*. Um dos gargalos na execução do método original é também a necessidade de observar gráficos e a partir deles tomar decisões relativas ao prosseguimento do processo. Como a execução dos diversos programas era feita em Linux e o programa plotador de gráficos, o Origin (ORIGIN, 2011), roda em Windows, em várias situações era necessário reiniciar a máquina ou mudar de máquina para que se pudesser plotar os gráficos. Com o Matplotlib (MATPLOTLIB, 2011), biblioteca de plotagem de gráficos para Python, esse trabalho passa a ser desnecessário.

O *BioPython* inclui o Blast2 e seus *bindings* para Python. Como módulos externos, se fizeram necessários ainda o Madchar (geração de matrizes), o RedeCrítica (análise de limiares) e o Dendo (clusterização), os três desenvolvidos pelo FESC. Por serem codificados em Fortran foi necessário o pacote Gfortran (GFORTRAN..., 2011). A Figura 4.1 e a Figura 4.2 ilustram as ferramentas utilizadas em cada etapa do processo e em relação às camadas do AIAF, respectivamente.

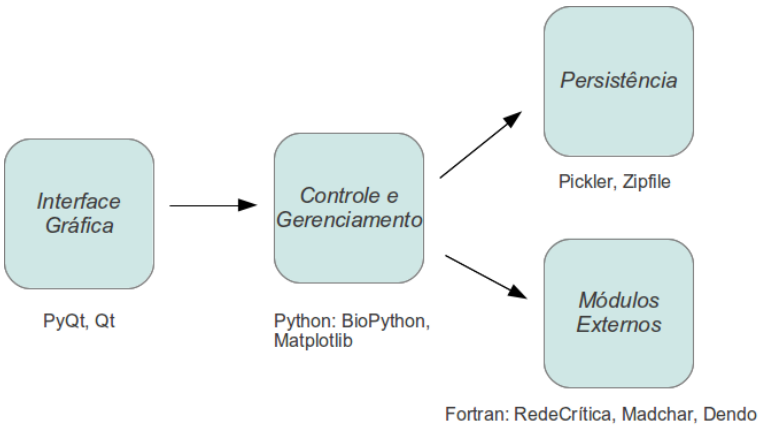

Figura 4.2: Ferramentas utilizadas na interface gráfica, camada de controle e gerenciamento, persistência e módulos externos.

4.2 APRESENTAÇÃO DO PROTÓTIPO

A primeira preocupação na implementação do AIAF foi a questão inicial da filtragem e armazenamento dos dados. Um banco de dados relacional é uma exigência muito grande para um sistema que vai realizar certos cálculos e gráficos, e que será utilizado em larga escala em máquinas diversas e por pesquisadores de áreas como Biologia. Por isso, a primeira decisão foi a troca do banco de dados relacional por uma estrutura de armazenamento de dados através da serialização de objetos, sendo útil também na criação de um projeto e o carregamento de um projeto existente no disco, que são atribuições do pacote *Persistence*. Por um lado se perde ao deixar de utilizar um sistema gerenciador de banco de dados, deixando a cargo do AIAF toda a responsabilidade sobre o correto armazenamento e recuperação dos dados, mas por outro exime o usuário da necessidade de instalar e configurar um ambiente de banco de dados relacional, o que exige um determinado nível de conhecimento técnico. Em sua versão atual, o protótipo do AIAF utiliza uma solução simples para a persistência, baseada no módulo *Pickler* do Python para a serialização de todos os objetos do sistema em um único arquivo.

Um arquivo .aiaf (extensão de arquivo do AIAF) é simplesmente um pacote compactado que contém uma estrutura de diretórios, destinada a armazenar os arquivos necessários à execução de um projeto, e também fornece suporte a certas operações de execução. Quando necessário, o pacote *Persistence* descompacta o .aiaf em um diretório temporário, espera certas operações serem realizadas e então compacta novamente na localização original, utilizando a biblioteca *zipfile* do Python. Os processos de compactação e descompactação são transparentes para o restante da aplicação.

A filtragem feita originalmente não incluía uma opção de escolha das sequências com que

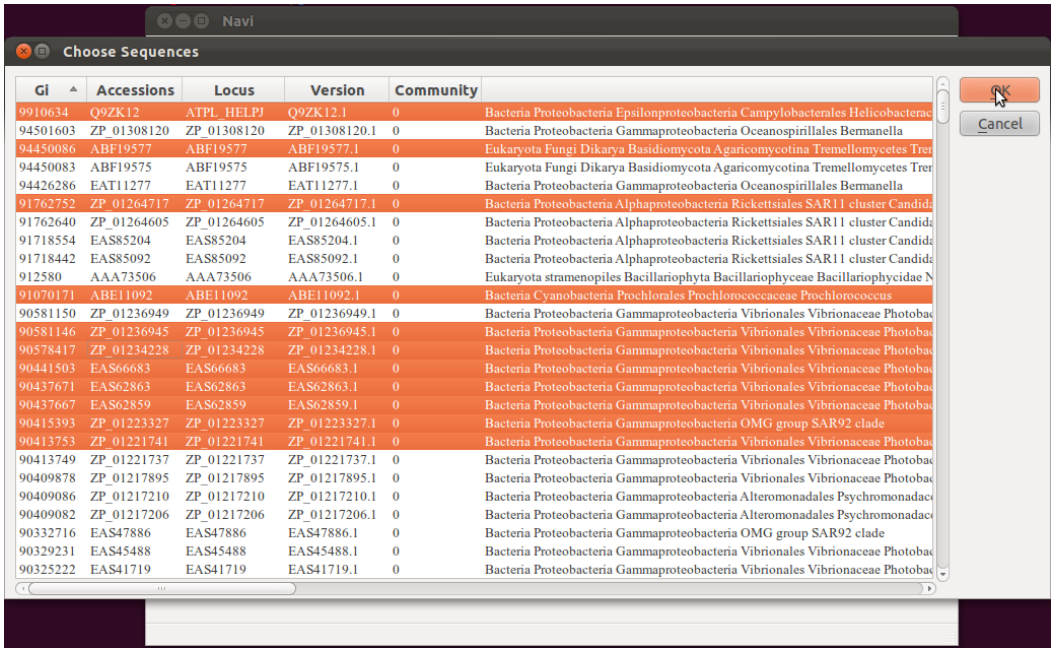

Figura 4.3: Escolha das sequências no AIAF após filtragem.

se desejasse trabalhar, fazendo com que todas as sequências dos arquivos do GenBank fossem utilizados na geração da matriz de similaridades. No AIAF, a criação de um novo projeto resulta na filtragem e armazenamento dos dados, e em seguida a possibilidade de escolha das sequências, como mostra a Figura 4.3.

A geração de matrizes de adjacência é feita na construção de um objeto da classe *AdjacencyMatrix*, sem a necessidade de invocar módulos externos, pela simplicidade da operação. Já para gerar as matrizes de vizinhança, o módulo Madchar é invocado. A operação também é realizada durante a construção de um objeto da classe *NeighbourhoodMatrix*, onde seu construtor invoca o método *generate_neighbourhood_values*. A Figura 4.4 ilustra a interface de geração de matrizes.

A visualização do gráfico de matriz de cores se baseia nos dados da matriz de vizinhança e simplesmente os utiliza como entrada para a biblioteca *Matplotlib* desenhar o gráfico na tela.

Como descrito na Seção 3.4, a classe *Analysis* tem a *ThresholdAnalysis* como subclasse, que por sua vez tem como descendentes classes que representam as diferentes formas de execução da análise de limiares. Isso permite uma extensibilidade ao AIAF, uma vez que basta acrescentar uma nova descendente à *ThresholdAnalysis* para se incluir um novo método de analisar limiares. A situação é análoga para a clusterização, com a classe *Clustering* e sua filha *NewmanGirvan*.

Nesta versão do protótipo do AIAF, apenas o método de distâncias foi implementado, por meio de invocação do módulo externo RedeCrítica por um método da classe *Distance*, que é

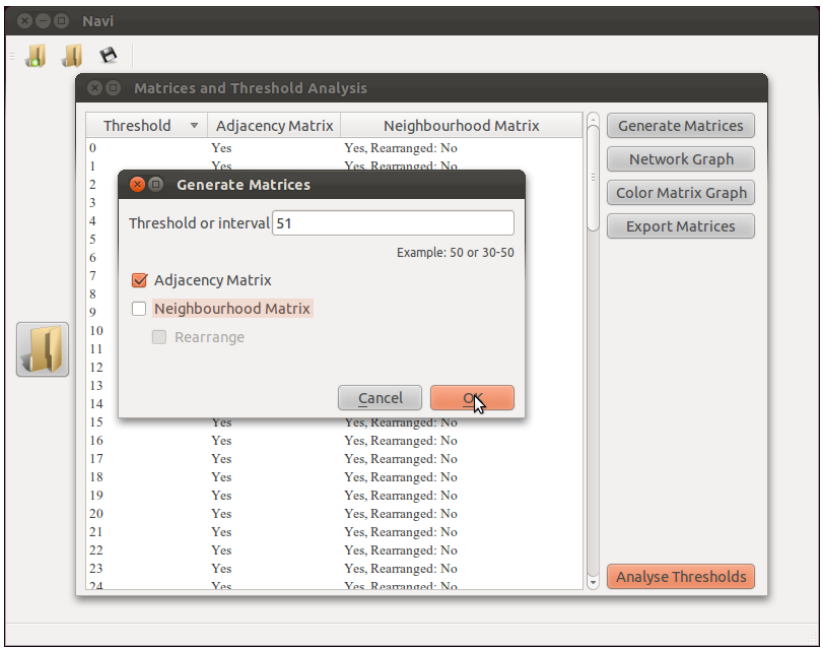

Figura 4.4: Geração de matrizes no AIAF.

filha da classe *ThresholdAnalysis*. Para a clusterização, o método de Newman e Girvan foi implementado por meio de invocação de um módulo externo feito em Fortran (Dendo). A invocação dos módulos Fortran foi feita utilizando chamadas de sistema da linguagem Python, que não permite uma integração perfeita, mas se utiliza do poder da linguagem de mapear as chamadas de sistema para a plataforma em que o *software* está sendo executado. A Figura 4.5 ilustra a interface com o usuário para a execução da clusterização e a Figura 4.6 ilustra o dendrograma, resultado final do método, gerado pelo AIAF.

A etapa de congruência foi modelada mas não foi implementada no protótipo pela dificuldade de implementação, pela necessidade de estudo profundo nos outros métodos de análise filogenética e as ferramentas existentes para tal, além de adequar as saídas a um formato padrão para executar a comparação. Tal trabalho exigiria mais tempo que o disponível. Mais informações sobre outros métodos e a congruência podem ser vistas em (DINIZ, 2010). Como o AIAF não é capaz de gerar gráficos tão bons quanto uma suíte profissional dedicada à plotagem destinada a artigos e trabalhos científicos em geral, e há perda do controle do usuário sobre a localização e o formato dos dados gerados entre as diversas etapas do processo, se faz necessária a exportação desses arquivos para que o usuário os utilize como fonte para utilização em programas plotadores.

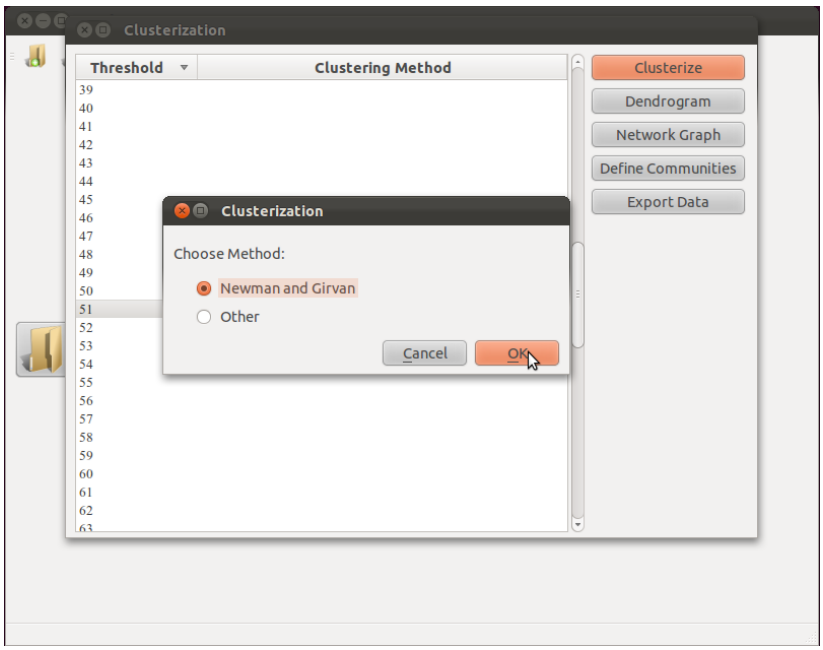

Figura 4.5: Interface de execução da clusterização no AIAF.

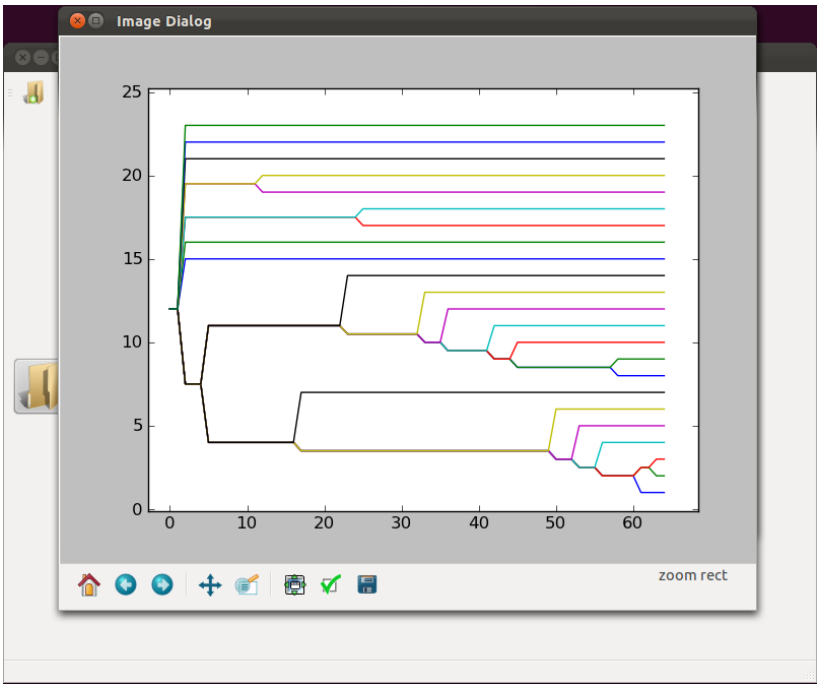

Figura 4.6: Dendrograma mostrado pelo AIAF.

4.3 DIFICULDADES

Uma das principais dificuldades, sem dúvida, foi conseguir pensar em um sistema que pudesse ser ao mesmo tempo extensível, garantisse uma facilidade de uso, se mostrasse rápido, de fácil manutenção, seguindo as boas práticas de modelagem e programação, sem perder a integração com o código legado. Tais características exigiram um pensamento integrado na modelagem, implementação e processo de desenvolvimento, sem fugir aos requisitos e aos interesses dos dois grupos de usuários: os pesquisadores e os desenvolvedores.

Ao longo do trabalho, houve complicações nas decisões de projeto e na modelagem. Na parte da implementação, a migração de alguns códigos originalmente em Perl para Python e a opção de abandonar um banco de dados relacional tomaram uma boa parte do tempo, como também o aprendizado de PyQt e o desenvolvimento da interface gráfica.

Apesar das dificuldades e da não implementação de alguns casos de uso, a modelagem realizada e o desenvolvimento do protótipo dentro do modelo em espiral consistiram um trabalho importante, cujas considerações finais estão dispostas no próximo Capítulo.

5 CONCLUSÃO

Este trabalho se utilizou de práticas de Engenharia de Software com o desafio de prover um ambiente para o controle do processo pelos usuários, e também oferecer aos desenvolvedores possibilidade de manutenção e extensibilidade, utilizando-se de uma organização arquitetural em camadas, sem a perda das características originais do método de análise filogenética através da integração com código legado, e incluindo a possibilidade de extensibilidade.

O primeiro protótipo do AIAF pode obter arquivos do GenBank, filtrá-los e a partir da escolha das sequências pelo usuário gerar a matriz de similaridades. A partir dela, permite ao usuário a geração de matrizes de adjacência e vizinhança, e em seguida analisar limiares e executar a clusterização. Além disso, possibilita a criação, carregamento e salvamento de projetos. O conjunto dessas funcionalidades constitui na finalização do primeiro ciclo do modelo em espiral.

O trabalho realizado aqui representa um ponto de partida para que não só um método possa ser executado, mas para que se tenha um ambiente muito maior, que possibilite a aplicação de outros métodos de análise filogenética, como também a comparação e a extração de dados estatísticos entre eles. A modelagem do ambiente permitirá a concepção de novas versões do protótipo, levando à continuação do modelo de desenvolvimento em espiral, e a modelagem pode ser posteriormente ampliada, incluindo novos requisitos e funcionalidades e aumentando seu escopo. A próxima seção listará o que pode ser feito como trabalhos futuros.

5.1 TRABALHOS FUTUROS

Os principais trabalhos futuros estão relacionados ao protótipo. Posteriormente, pode-se ampliar o escopo e consequentemente a modelagem. Seguem os trabalhos futuros:

- Dois casos de uso importantes que não foram implementados no protótipo são a definição de módulos/comunidades (caso de uso 7), onde o usuário define com base na matriz de cores e no dengrograma quais sequências pertencem a quais comunidades e com isso pode ser gerado o gréfico completo da rede (caso de uso 9) para visualização, onde o usuário

pode ver as comunidades separadas por cores, além de obter facilmente informações sobre as sequências clicando nos vértices. Estas funcionalidades em conjunto exigem um trabalho de interface gráfica grande, incluindo a exibição interativa de gráficos e a integração de bibliotecas de exibição de gráficos a bibliotecas de plotagem.

- Outros casos de uso também não foram implementados no protótipo, principalmente por questões de tempo: comparar/executar congruência de redes de diferentes projetos (caso de uso 8), executar congruência em relação a um arquivo cadastrado em um formato padrão (caso de uso 10) e exportar dados do projeto para gráficos em um formato de programas plotadores (caso de uso 11);

- A validação do protótipo é uma etapa muito importante neste processo de desenvolvimento e não foi realizada por conta de tempo e também pelo fato de o mesmo não apresentar maturidade suficiente. Após a implementação de alguns casos de uso e soluções de questões de interface gráfica e comunicação com o usuário como veremos abaixo, sua validação torna-se imprescindível;

- Fazer uma modelagem bem-definida do pacote de persistência;

- Melhorar o armazenamento e serialização de objetos: a persistência foi tratada como pacote na modelagem pela sua tendência de variação conforme a plataforma. Na implementação foi criado um modelo simples de persistência, a partir da serialização dos objetos do sistema em um único arquivo. Ao carregar um projeto, o sistema lê esse arquivo e recupera o estado original;

- O controle do pacote de persistência ficou centralizado na classe *Project*. É necessário uma análise sobre as vantagens dentro do projeto de tornar este controle distribuído;

- Execução remota de algoritmos: alguns métodos de clusterização levam alguns dias para executar. Muitas vezes é interessante executá-lo em uma outra máquina de forma remota. É interessante haver uma interface ssh (Secure Shell) para a execução remota de algumas análises, à escolha do usuário;

- A interface gráfica, apesar de representar uma melhoria em termos de uso, precisa ser melhorada com a criação de *widgets* em substituição às janelas *popup*, utilizando melhor os recursos do Qt;

- O protótipo não avisa ao usuário sobre algumas operações que estão ocorrendo e nem o progresso de algumas análises, o que é algo importante pois determinadas análises

demoram muito tempo para serem realizadas, tornando esse *feedback* para o usuário algo essencial;

- Uso de *threads*: algumas análises podem ser realizadas com o uso de *threads* para liberar a interface gráfica, para o usuário possa realizar outras operações no sistema enquanto alguma análise estiver em andamento;

- A integração do sistema (feito em Python) com os módulos em Fortran está utilizando chamadas do sistema operacional. Python oferece um *binding* para Fortran que poderia ser utilizado, pois proporcionaria a integração direta entre as duas linguagens sem utilizar chamadas ao sistema operacional como uma espécie de camada intermediária para a comunicação entre elas.

APÊNDICE A – DIAGRAMA DE CLASSES COMPLETO

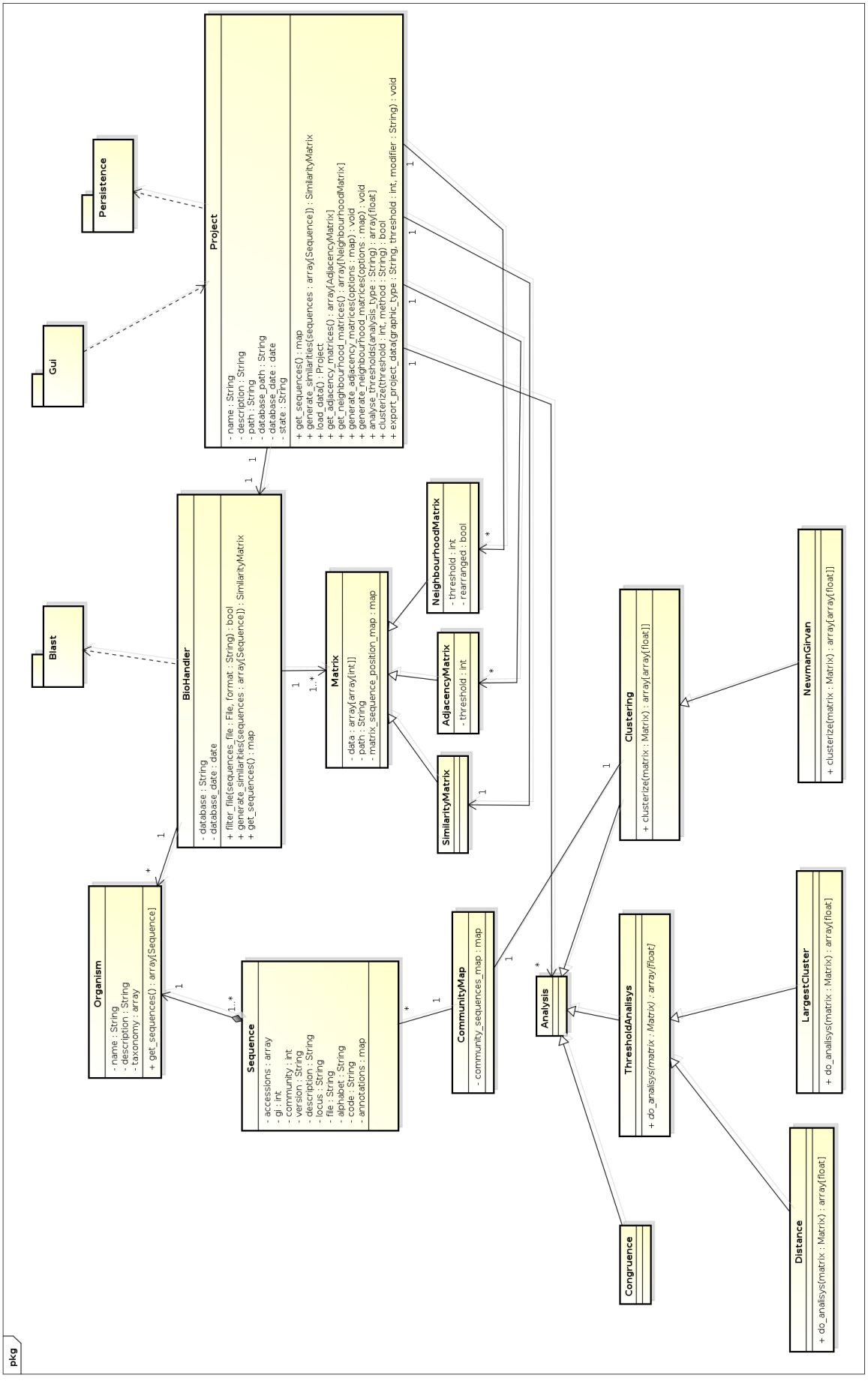

Figura A.1: Diagrama de classes completo do AIAF.

APÊNDICE B – DIAGRAMAS DE SEQUÊNCIAS

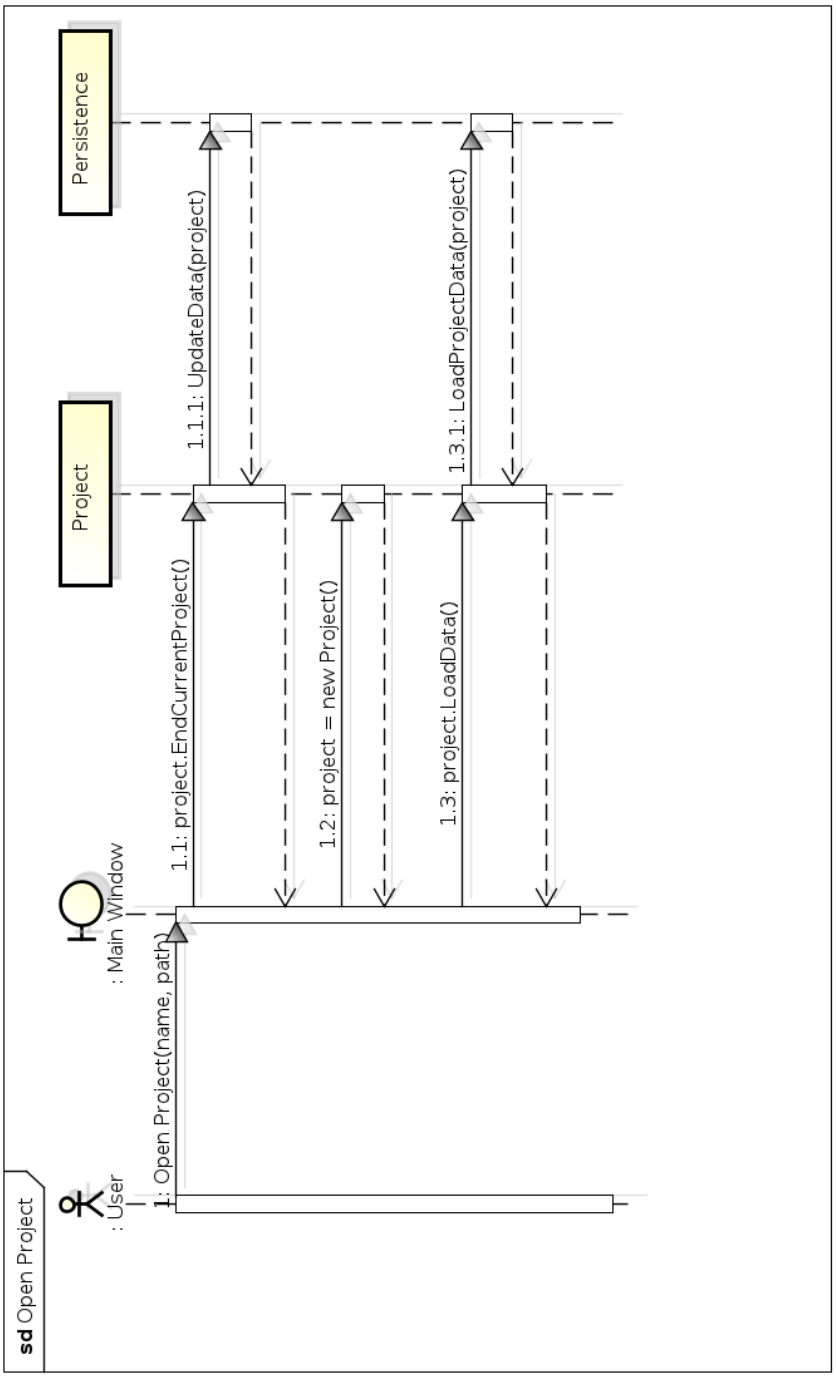

Figura B.1: Diagrama de sequências para a abertura de projeto existente referente ao caso de uso 1: gerenciar projeto.

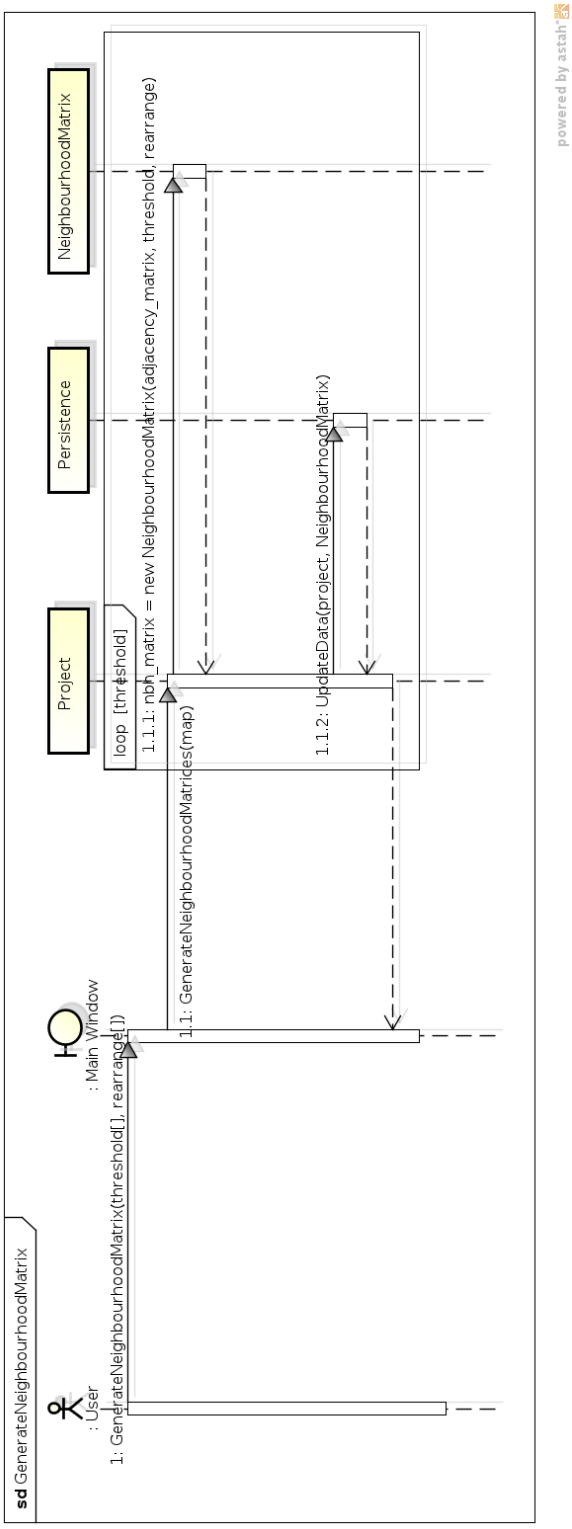

Figura B.2: Diagrama de sequências para o caso de uso 3: gerar matriz de vizinhança de um limiar qualquer.

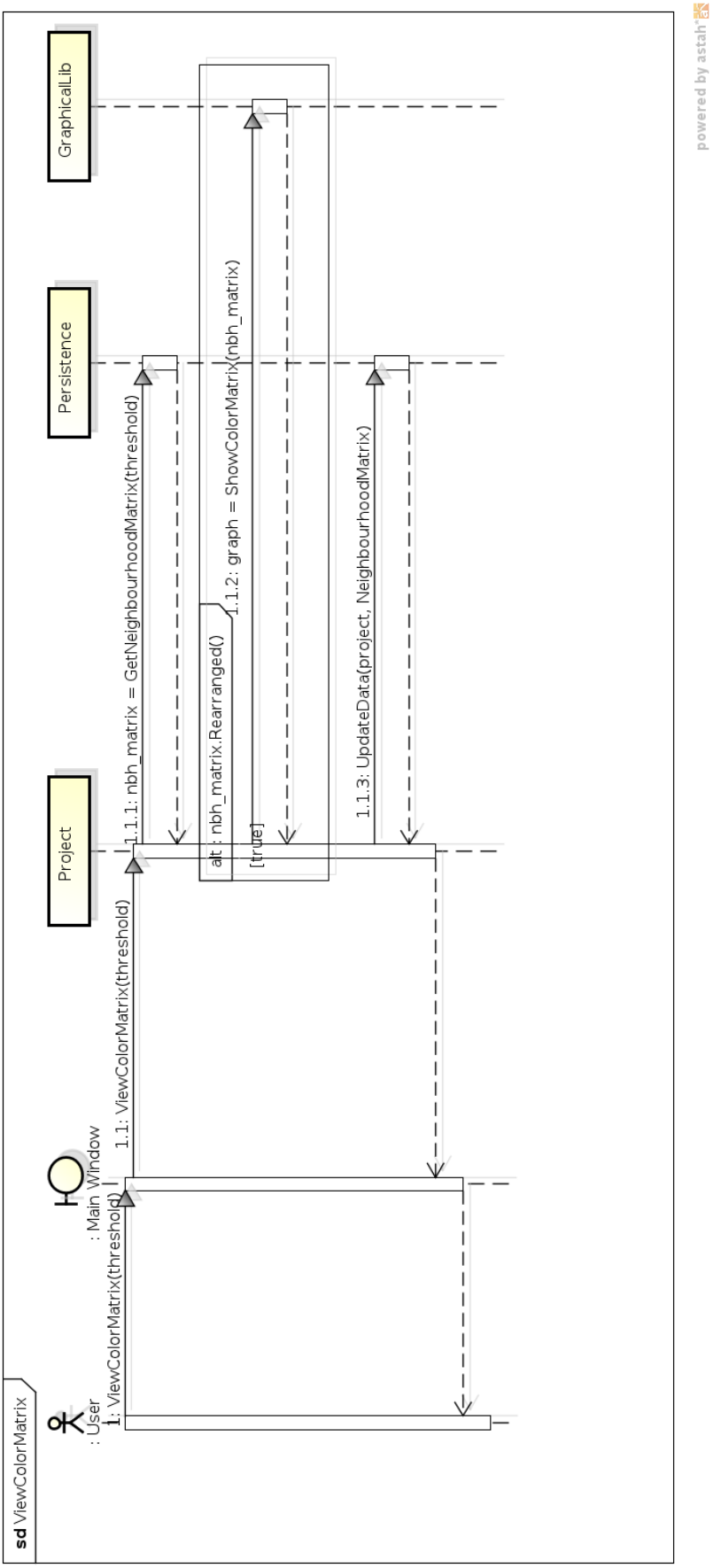

Figura B.3: Diagrama de sequências para o caso de uso 4: visualizar gráfico de matriz de vizinhança qualquer em forma de matriz de cores.

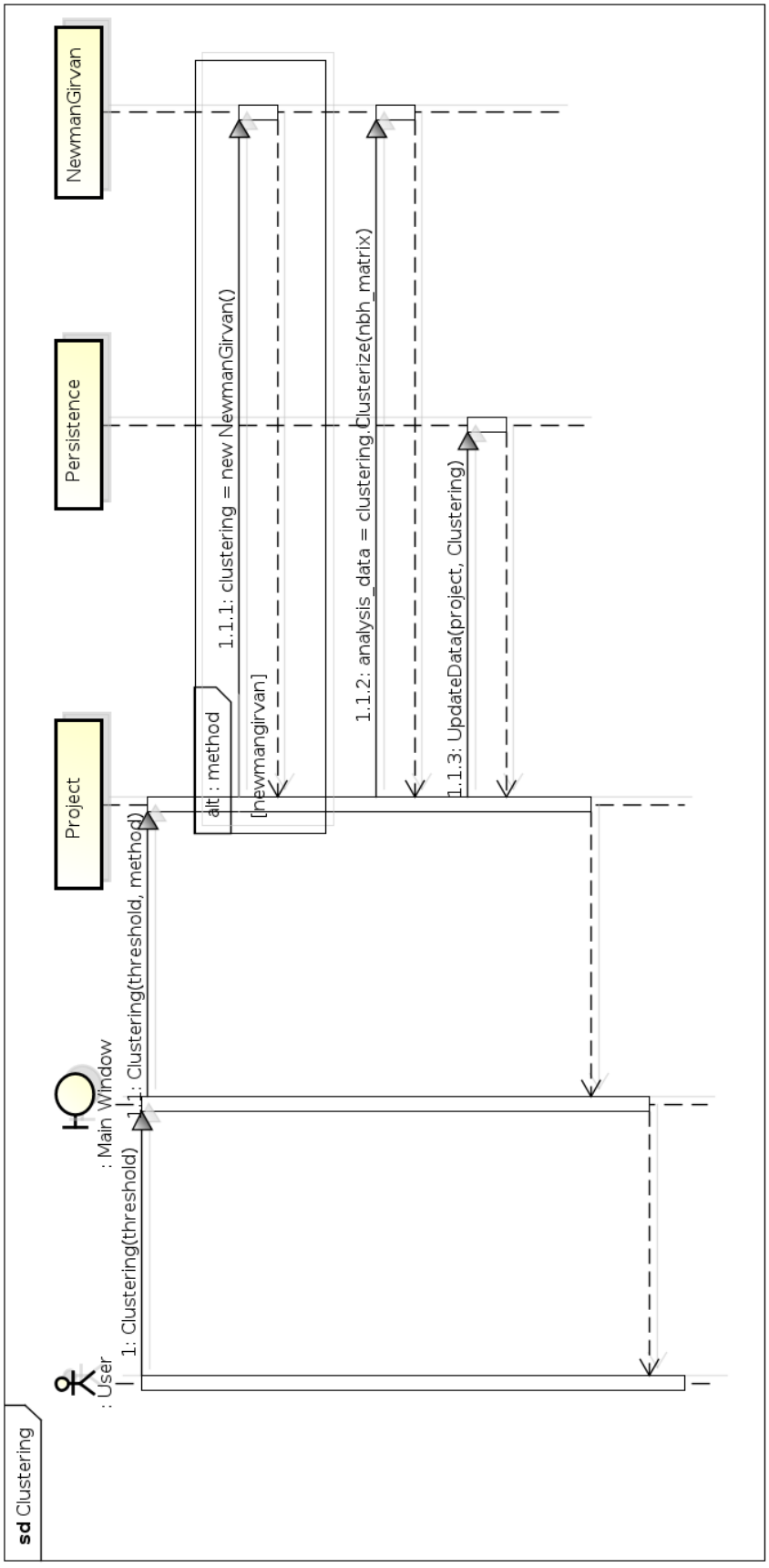

Figura B.4: Diagrama de sequências para o caso de uso 6: executar clusterização.

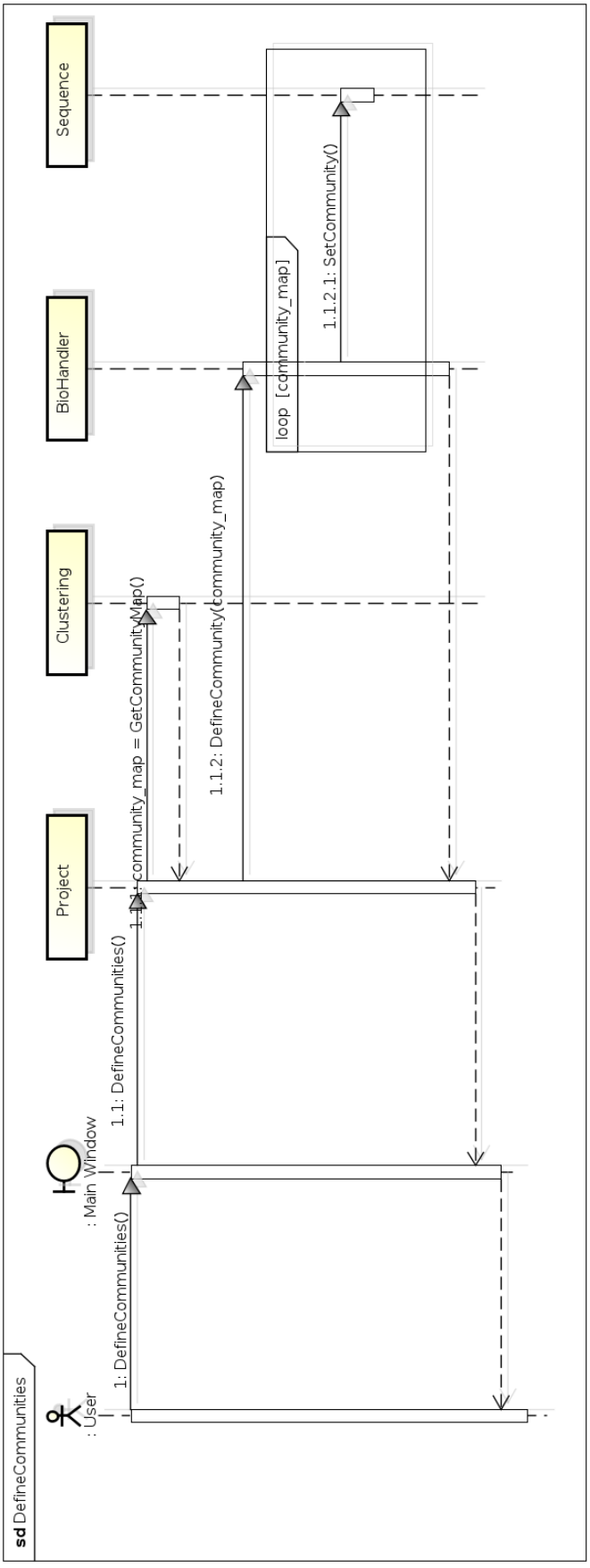

Figura B.5: Diagrama de sequências para o caso de uso 7: definir módulos/comunidades.

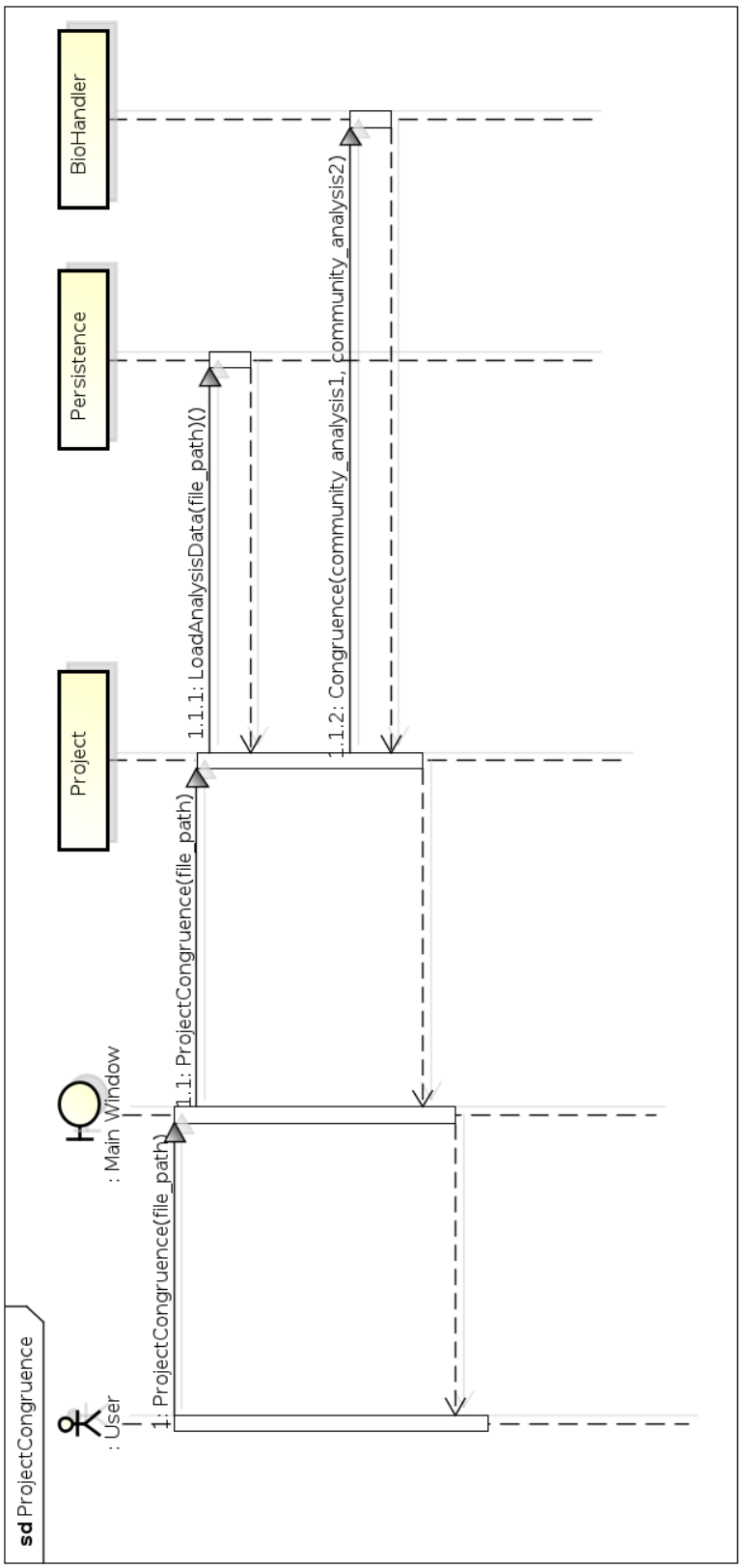

Figura B.6: Diagrama de sequências para o caso de uso 8: comparar/executar congruência de redes de diferentes projetos.

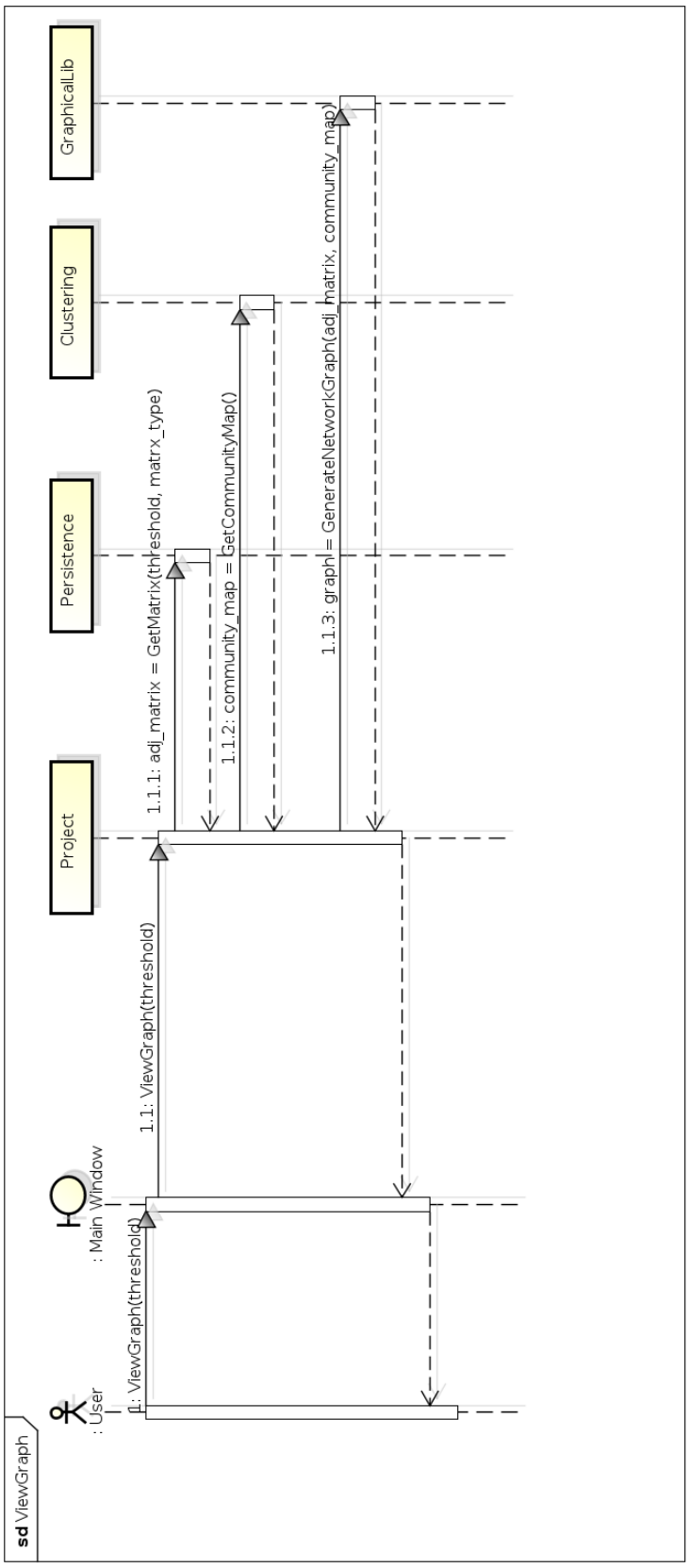

Figura B.7: Diagrama de sequências para o caso de uso 9: visualizar gráfico completo da rede.

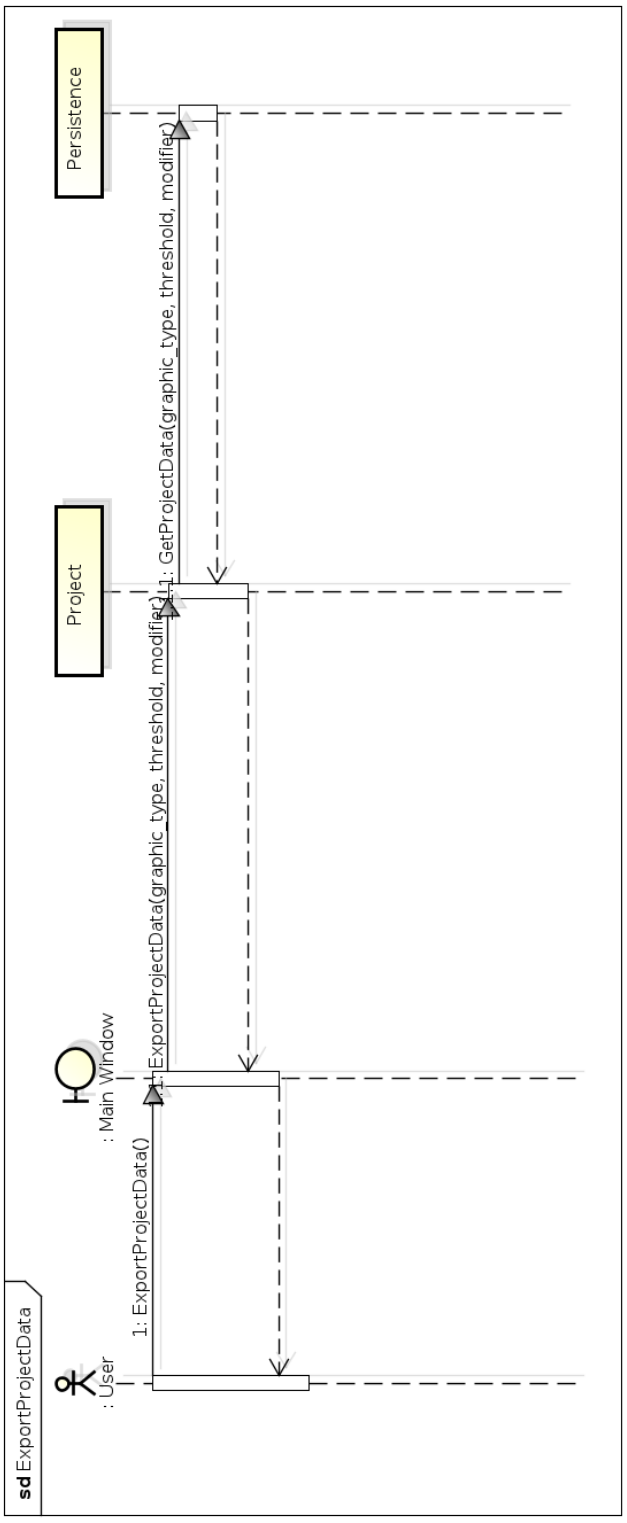

Figura B.8: Diagrama de sequências para o caso de uso 11: exportar dados do projeto para gráficos.

REFERÊNCIAS BIBLIOGRÁFICAS

ALTSCHUL, S. et al. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res.*, v. 25, p. 3389–3402, 1997.

ANDRADE, R. F. S.; MIRANDA, J. G. V.; LOBÃO, T. P. Neighborhood properties of complex networks. *Phys. Rev. E.*, v. 73, 2006.

ANDRADE, R. F. S. et al. Measuring distances between complex networks. *Phys. Lett. A.*, v. 372, p. 5265–5269, 2008.

ANDRADE, R. F. S. et al. Detecting network communities: An application to phylogenetic analysis. *PLoS Computational Biology.*, v. 7, p. e1001131, 2011.

ANDRADE, R. F. S.; PINHO, S. T.; LOBÃO, T. P. Identification of community structure in networks using higher order neighborhood concepts. *Int. J. Bifurc. Chaos 19.*, v. 8, 2009.

ASTAH Community. 2011. Último acesso em 05 de Novembro de 2011. Disponível em: <http://astah.net/editions/community>.

BARABASI, A. L.; OLTVAI, Z. N. Network biology: understanding the cell's functional organization. *Nat. Rev. Genet.*, v. 5, p. 101–130, 2004.

BESSA, A. D.; SANTOS, L. B. L.; COSTA, M. C. Introdução às redes complexas. *pre-print.*, 2008.

BIOPYTHON. 2011. Último acesso em 05 de Novembro de 2011. Disponível em: <http://biopython.org/>.

BOEHM, B. A spiral model of software development and enhancement. *ACM SIGSOFT Software Engineering Notes.*, v. 11, n. 4, p. 14–24, 1986.

DIESTEL, R. *Graph Theory*. [S.l.]: Springer-Verlag, 2010.

DINIZ, M. V. C. *Análise Computacional de sistases da quitina de fungos basidiomicetos*. Dissertação (Mestrado) — Universidade Estadual de Feira de Santana, Feira de Santana, 2010.

GFORTRAN - GCC Wiki. 2011. Último acesso em 05 de Novembro de 2011. Disponível em: <http://gcc.gnu.org/wiki/GFortran>.

GÓES-NETO, A. et al. Comparative protein analysis of the chitin metabolic pathway in extant organisms: A complex network approach. *Biosystems (Amsterdam. Print).*, v. 101, p. 59–66, 2010.

MATPLOTLIB. 2011. Último acesso em 05 de Novembro de 2011. Disponível em: <http://matplotlib.sourceforge.net/>.

NATIONAL Center for Biotechnology Information. 2011. Último acesso em 05 de Novembro de 2011. Disponível em: <http://www.ncbi.nlm.nih.gov/>.

NEWMAN, M. E. J.; GIRVAN, M. Finding and evaluating community structure in networks. *Phys. Rev. E*, v. 69, 2004.

NIELSEN, J. *Usability Engineering*. [S.l.]: Morgan Kaufmann, 1993.

ORIGIN. 2011. Último acesso em 05 de Novembro de 2011. Disponível em: <http://www.originlab.com/>.

PRESSMAN, R. S. *Software Engineering: A Practitioner's Approach*. [S.l.]: McGraw-Hill, 2005.

PYQT Reference Guide. 2011. Último acesso em 05 de Novembro de 2011. Disponível em: <http://www.riverbankcomputing.co.uk/static/Docs/PyQt4/html/index.html>.

PYTHON Programming Language – Official Website. 2011. Último acesso em 05 de Novembro de 2011. Disponível em: <http://www.python.org/>.

QT - A cross-platform application and UI framework. 2011. Último acesso em 05 de Novembro de 2011. Disponível em: <http://qt.nokia.com/products/>.

SCHNEIDER, H. *Métodos de Análise Filogenética - Um Guia Prático*. [S.l.]: Holos, 2007.

THE Gtk+ Project. 2011. Último acesso em 05 de Novembro de 2011. Disponível em: <http://www.gtk.org/>.

UML. 2011. Último acesso em 05 de Novembro de 2011. Disponível em: <http://www.omg.org/spec/UML/2.4/>.

GLOSSÁRIO

artefatos computacionais são vários tipos de subprodutos concretos produzido durante o desenvolvimento de software. Alguns artefatos (por exemplo, casos de uso, diagramas de classes e outros modelos UML, requisitos e documentos de projeto) ajudam a descrever a função, arquitetura e o design do software. Outros artefatos estão relacionados com o próprio processo de desenvolvimento - tais como planos de projetos, processos de negócios e avaliações de risco. Podem ser manuais, arquivos executáveis, módulos etc. 7

Bioinformática corresponde a aplicação das técnicas da Informática, no sentido de análise da informação na área de estudo da Biologia, utilizando tais técnicas computacionais e matemáticas à geração e gerenciamento de bioinformação. 7

Congruência trata da comparação entre duas redes de sequências proteicas, que é feita após a definição de módulos/comunidades de cada uma das redes. Nela analisa-se a porcentagem de sequências proteicas que pertencendo a uma determinada comunidade da primeira rede, também estão presentes na comunidade correspondente na segunda rede. 18

Diagrama de Casos de Uso descreve a funcionalidade proposta para um novo sistema que será projetado. Pode ser tido como um documento narrativo que descreve a sequência de eventos de um ator que usa um sistema para completar um processo. 9

Diagrama de Classes é uma representação da estrutura e relações das classes que servem de modelo para objetos. É uma modelagem muito útil para o sistema, define todas as classes que o sistema necessita possuir e é a base para a construção dos diagramas de comunicação, sequência e estados. 9

Diagrama de Sequências descreve a maneira como os grupos de objetos colaboram em algum comportamento ao longo do tempo. Ele registra o comportamento de um único caso de uso e exibe os objetos e as mensagens passadas entre esses objetos no caso de uso. 9

Engenharia de Software área do conhecimento da computação voltada para a especificação, desenvolvimento e manutenção de sistemas de software aplicando tecnologias e práticas

de gerência de projetos e outras disciplinas, objetivando organização, produtividade e qualidade. 7

Entremeação ou *Betweenness*, é uma medida de centralidade de um vértice ou aresta de uma rede. Nesta concepção, vértices ou arestas que tenham alta probabilidade de serem percorridas na escolha aleatória de um caminho mínimo entre dois vértices da rede possuem uma alta entremeação. 14

Física Estatística é o ramo da Física que usa métodos da teoria das probabilidades e estatística ferramentas matemáticas e computacionais para lidar com sistemas físicos formados por um número muito grande de elementos. Seu principal objetivo é esclarecer as propriedades globais, com base no comportamento médio de seus constituintes. Nos últimos anos técnicas da Física Estatística têm sido utilizadas para estudar problemas de outras áreas de conhecimento (Biologia, Economia, etc) que lidam com sistemas formados por um número muito grande de elementos. 7

GenBank é um formato de exibição de sequências, consistindo em uma seção de anotações contendo locus, identificador, palavras-chave, definição, organismo, entre outras informações; e uma seção de sequência, que contém o código da sequência propriamente dita. O fim de uma seção é demarcado por "//". 11

matriz de cores é a matriz de vizinhança representada de forma que cada número recebe uma cor diferente, variando do azul ao vermelho à medida em que esse número aumenta. A matriz de vizinhança pode ser rearranjada de forma a agrupar números iguais com a maior proximoidade possível da diagonal principal, resultando em um gráfico de matriz de cores que permite uma melhor interpretação a partir da visualização. 21

matrizes de vizinhança são matrizes derivadas das matrizes de adjacência, onde, no lugar dos números 0 referentes a uma não conexão entre o par de vértices em questão, tem-se um número natural, representando o caminho mínimo entre os vértices. 12

modelagem atividade de construir modelos que expliquem as características ou o comportamento de um *software*. Na construção do *software* os modelos podem ser usados na identificação das características e funcionalidades que o *software* deverá prover (análise de requisitos), e no planejamento de sua construção. 8

método das distâncias método de comparação de matrizes de vizinhança de limiares consecutivos através da distância euclidiana entre elas, ou seja, pela soma das diferenças em módulo das posições da mesma. 13

persistência refere-se ao armazenamento não-volátil de dados, por exemplo, o armazenamento em um dispositivo físico como um disco rígido. Pode-se dizer que de maneira geral, o termo persistência é associado a uma ação que consiste em manter em meio físico recuperável, como banco de dados ou arquivo, de modo a garantir a permanência das informações de um determinado estado de um objeto lógico.. 8

Sistemas Complexos são sistemas onde suas propriedades não são uma consequência natural de seus elementos constituintes vistos isoladamente. As propriedades emergentes de um sistema complexo decorrem em grande parte da relação não-linear entre as partes. 7