Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Incremental Learning de Redes Bayesianas e o UnBBayes

Pedro da C. Abreu

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Marcelo Ladeira

Coorientador
Prof. Dr. Shou Matsumoto

Brasília
2015

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Incremental Learning de Redes Bayesianas e o UnBBayes

Pedro da C. Abreu

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Marcelo Ladeira (Orientador)
CIC/UnB

Prof. Dr. Shou Matsumoto
CIC/GMU

Prof. Dr. Laecio Lima
CIC/UnB

Prof. Dr. Homero Luiz Piccolo
Coordenador do Bacharelado em Ciência da Computação

Brasília, 07 de dezembro de 2015

Resumo

Com o crescente interesse em aprendizado de máquina, frameworks robustos e implementações do estado da arte são uma necessidade do mundo moderno. Pensando nisso levantamos um estudo sobre o estado da arte de aprendizado no contexto de Rede Bayesiana (BN), já que BNs provêm um excelente modelo para lidar com relações causais e probabilidade bayesiana. Para tal utilizaremos o framework UnBBayes e desenvolveremos plugins para este.

Palavras-chave: LaTeX, metodologia científica, trabalho de conclusão de curso

Abstract

With the growing interest in machine learning, good frameworks and state of the art implementation are a need in modern world. With this in mind we provide a study in the state of the art of BN, since BNs provide a great model to deal with causal relations and bayesian probability. For such we shall use UnBBayes framework and implement plugins for it.

Keywords: LaTeX, scientific method, thesis

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

API Aplication Program Interface.

BN Bayesian Network.

CPT Conditional Prability Tables.

DAG Directed Acyclic Graph.

GIA Grupo de Inteligencia Artificial.

GMU George Mason University.

GOF Gang of Four.

GUI Graphical User Interface.

ID Influence Diagram.

MDI Multiple Document Interface.

MEBN Multi Entity Bayesian Network.

MSBN Multiple-Sectioned Bayesian Network.

MVC Model View Controller.

OO Orientado a Objetos.

PR-OWL Probabilistic Web Ontology Language.

UnB Universidade de Brasília.

Capítulo 1

Introdução

No cotidiano do ser humano é muito comum raciocinar e tomar decisões sob condições de incerteza. Isto é tão visível e profundo para algumas pessoas que no século XVIII Bishop Butler declarou "probabilidade é o guia da vida".

Como um exemplo do quanto probabilidade é importante tome a medicina. Para um especialista médico determinar a doença de uma pessoa com base nos sintomas observados (evidências) é preciso que ele leve em conta a probabilidade daquele sintoma refletir esta ou aquela doença, pois a doença ocasiona determinado sintoma apenas com alguma probabilidade, mas não com certeza.

Podemos ainda citar vários outros exemplos, como na economia, teoria dos jogos, genética, previsão do tempo.

Pensando nisso é necessário um modelo robusto para lidar com tantas incertezas, para tanto escolhemos as BNs.

1.1 Motivação

BN é um modelo gráfico para relações probabilísticas dado conjunto de variáveis. Nas últimas décadas, redes Bayesianas se tornaram representações populares para codificar conhecimento incerto para sistemas especialistas [7]. Mais recentemente, pesquisadores desenvolveram métodos de aprendizagem de redes Bayesianas a partir de dados. As técnicas desenvolvidas são relativamente novas e ainda em evolução, mas eles têm se mostrado muito eficientes para alguns problemas de análise de dados.

Existem diversos representações possíveis para analise de dados, entre elas, decision tres, e redes neurais artificiais; e outras tantas como estimação de densidade, classificação, regressão e clusterings. Portanto o que métodos de BNs têm a oferecer? Segundo Heckerman [7] podemos oferecer pelo menos quatro respostas, sendo elas:

1. BNs lidam com um conjunto incompleto de dados de maneira natural.

2. BNs permitem aprender sobre as relações causais. Aprender sobre tais relações são importantes por pelo menos duas razões: O processo é útil quando se está tentando entender sobre um dado problema de domínio, como por exemplo, durante uma análise de dados exploratória. E mais, conhecimento de relações causais nos permitem fazer predições na presença de intervenções. Por exemplo, um analista de mercado pode querer saber se é lucrativo aumentar o investimento em determinada propaganda para aumentar as vendas de seu produto. Para responder esta pergunta o analista pode determinar se esta propaganda é a causa para o aumento de suas vendas, e em caso afirmativo, quanto. O uso de BNs nos ajuda a responder tal pergunta até mesmo quando não há experimentos nos efeitos de tal propaganda.

3. BNs em conjunto com técnicas estatísticas bayesianas facilitam a combinação de conhecimento de domínio e dados. Qualquer um que tenha feito uma análise do mundo real sabe a importância de conhecimento prévio ou de domínio, em especial quando os dados são poucos ou caros. Pelo fato de alguns sistemas comerciais (i.e., sistemas especialistas) podem ser construídos a partir de conhecimentos prévios. BNs possuem uma semântica causal que permitem conhecimentos prévios serem representados de uma forma muito simples e natural. Além disto, BNs encapsulam tais relações causais com suas probabilidades. Consequentemente, conhecimento prévio e dados podem ser combinados com técnicas bem estudadas da estatística Bayesiana.

4. Métodos Bayesianos em conjunto com BNs e outros tipos de modelos oferecem uma forma eficiente para evitar over fitting dos dados. Como veremos, não há necessidade de excluir parte dos dados do treinamento do aprendizado da rede. Usando técnicas Bayesianas, modelos podem ser "suavizados"de tal forma que todo dado disponível pode ser usado para o treinamento.

1.2 Objetivos

O objetivo deste trabalho é apresentar os conhecimentos adquiridos durante este semestre na disciplina de Estudos Em Inteligência Artificial com o professor Ladeira da Universidade de Brasília (UnB), com a colaboração do Doutorando Shou Matsumoto da George Mason University (GMU), através de aulas pelo skype. E também firmar bases sólidas sobre os conhecimentos que serão necessários na concepção da verdadeira monografia, isto é, o trabalho de conclusão de curso.

1.3 Estrutura da Monografia

Este documento é organizado da seguinte maneira: no capítulo 2 explicamos a visão probabilística necessária, isto é, firmamos a base do trabalho que se segue, como todos axiomas e definições necessários. No capítulo 3 formalizamos o conceito de BNs e algumas de suas propriedades. No capítulo 4 discutimos como construí-las a partir de um conjunto de dados. No capítulo 5 explicamos o framework do UnBBayes, como ele funciona e como extendê-lo. O capítulo 5 discutimos alguns dos Design Patterns mais importantes. No capítulo 6 resumimos o trabalho do semestre e oferecemos uma prévia dos trabalhos a serem feitos nos próximos semestres.

Capítulo 2

Probabilidade

2.1 Probabilidade física versus probabilidade Bayesiana

Para entender BNs e as técnicas de aprendizado associadas, é importante entender a diferença entre a Probabilidade e Estatística padrão e a Bayesiana [9]

A Probabilidade física, também conhecida como probabilidade frequentista, foi advogada pelo matemático John Venn [16] no século XIX, identificando probabilidade com frequências dos eventos no longo prazo. A reclamação mais óbvia que surge neste modelo é que as frequências no curto prazo obviamente não casam com as calculadas, por exemplo, se jogarmos a moeda apenas uma vez, certamente concluiríamos que a probabilidade de cara é ou 1 ou 0.

Uma alternativa ao conceito de probabilidade física é pensar nas probabilidades como nosso grau de crença subjetivo. Esta visão foi expressa por Thomas Bayes [2] e Pierre Simon de Laplace [10] a duzentos anos atrás. Esta é uma visão mais geral da probabilidade pois leva em conta que temos crenças subjetivas em um grande variedade de proposições, muitas das quais não estão claramente ligadas a um processo físico. Por exemplo, a maioria de nós acreditamos na Hipótese de Copérnico de que a terra orbita em torno do sol, mas isto é baseado em evidencia não obviamente da mesma forma que um processo de amostragem. Isto é, ninguém é capaz de gerar sistemas solares repetidamente e observar a frequência com a qual os planetas giram em torno do sol. Seja como for Bayesianistas estão preparados para conversar sobre a probabilidade da tese de Copérnico ser verdadeira e ainda relacionar as relações de evidências a favor e contra ela.

Bayesiansmo pode ser visto como uma generalização da probabilidade física. Para isto adotamos o que David Lewis apelidou de Principal Principle [13]: sempre que se aprender uma probabilidade física de uma amostragem r, atualize sua probabilidade subjetiva para

aquela amostragem r. Basicamente isto é senso comum: pense que para você a probabilidade de um colega raspar a cabeça como 0.01, mas se você aprende que ele faz isso se e somente se um dado justo for lançado e der 2, você certamente revisará sua opinião de acordo.

Tendo explicado como as probabilidades Físicas e Bayesianas são compatíveis podemos seguir para a definição dos axiomas da probabilidade.

2.2 Axiomas da Probabilidade

Seja U o universo de possíveis evento. Os três Axiomas de Kolmogorov [8] afirmam que:

Axioma 2.1 *A probabilidade do acontecimento certo U, é 1:*

$$P(U) = 1$$

Axioma 2.2 *A probabilidade de qualquer acontecimento é maior ou igual a zero*

$$Para\ todo\ X \subseteq U, P(X) \geq 0$$

Axioma 2.3 *Dados dois acontecimentos disjuntos, a probabilidade da sua união é igual à soma das probabilidades de cada um*

$$Para\ todo\ X, Y \subseteq U,\ se\ X \cap Y = \emptyset,\ então\ P(X \cup Y) = P(X) + P(Y)$$

Definição 2.1 ***Probabilidade Condicional***

$$P(X|Y) = \frac{P(X \cap Y)}{P(Y)}$$

A probabilidade condicional exprime a seguinte ideia: dado que o evento Y já ocorreu, ou vai ocorrer, a probabilidade de X também ocorrer é $P(X|Y)$

Definição 2.2 ***Independência***

$$X \perp Y \equiv P(X|Y) = P(X)$$

Dois eventos X e Y são probabilisticamente independentes se, ao condicionar sobre um, o outro permanece igual.

2.3 Teorema de Bayes

Teorema 2.1 ***Teorema de Bayes***

$$P(h|e) = \frac{P(e|h)P(h)}{P(e)}$$

Este teorema diz que a probabilidade de uam hipótese h condicionada sobre alguma evidencia e é igual à multiplicação da crença a priori $P(h)$ pela verossimilhança $P(e|h)$. Desta forma $P(e|h)$ é chamada de probabilidade a posteriori.

É importante observar que $P(h|e) + P(\neg h|e) = 1$ e isto implica que $P(e|h)P(h) + P(e|\neg h)P(\neg h) = P(e)$

Capítulo 3

Redes Bayesianas

Uma BN provê uma representação compacta de distribuições de probabilidades grandes demais para lidar usando especificações tradicionais e provê um método sistemático e localizado para incorporar informação probabilística sobre uma dada situação.

Uma BN é um Directed Acyclic Graph (DAG) que representa uma função de distribuição de probabilidades conjunta de variáveis que modelam certo domínio de conhecimento. Ela é constituída de uma DAG, de variáveis aleatórias (também chamadas de nós da rede), arcos direcionados da variável pai para a variável filha e uma Conditional Prability Tables (CPT) associada a cada variável.

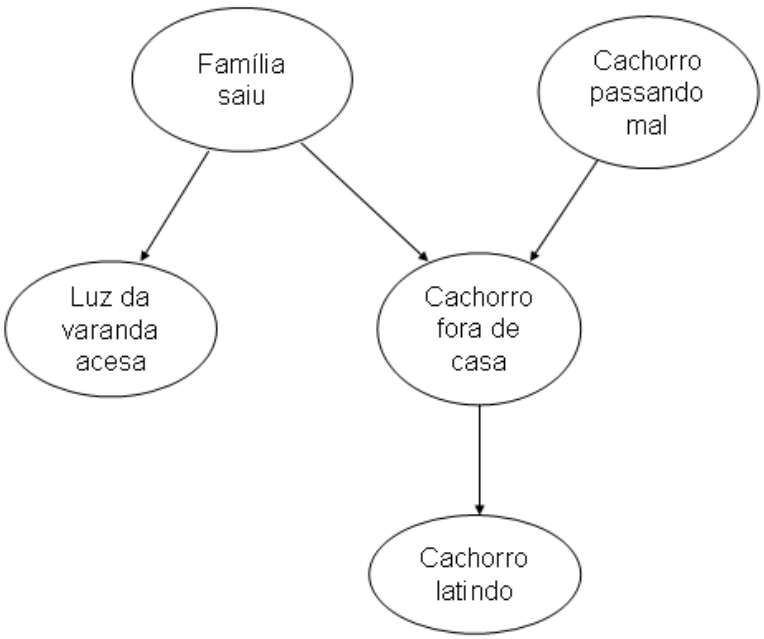

Figura 3.1: Exemplo family-out

Nesse exemplo, suponhamos que se queira determinar se a família está em casa ou se

ela saiu. Pelo grafo, podemos perceber que o fato de a luz da varanda estar acesa e de o cachorro estar fora de casa são indícios de que a família tenha saído.

3.1 Definição Formal

Considere a BN contendo n nós, X_1 até X_n, tomados nesta ordem. Uma instanciação da distribuição disjunta de probabilidade é representada por $P(X_1 = x_1, X_2 = x_2, ..., X_n = x_n)$, ou de forma compacta $P(x_1, x_2, ..., x_n)$. A **regra da cadeia** nos permite fatorar probabilidades disjuntas como:

$$P(x_1, x_2, ..., x_n) = P(x_1)P(x_2|x_1)...P(x_n|x_1, .., x_{n-1}) = \prod_i P(x_i|x_1, ..., x_{i-1}) \tag{3.1}$$

No entanto, a estrutura de uma BN implica que o valor de um nó em particular é condicionado apenas aos valores dos nós pais, o que reduz a equação acima à:

$$P(x_1, x_2, ..., x_n) = \prod_i P(x_i|pa_i) \tag{3.2}$$

Onde pa_i são os nós pais de x_i

3.2 D-separation

Segundo Cheng[3] Uma rede bayesiana pode ser vista como um sistema de rede de canais de informação, onde cada não é uma válvula que está aberta ou fechada e as válvulas são conectadas por canais de informação ruidosos (arestas). O fluxo de informação pode passar por um válvula aberta, mas não por uma válvula fechada. Quando todas as válvulas sobre um caminho entre dois não estão abertas, diz-se que este caminho é aberto. Se qualquer válvula no caminho está fechada, diz-se que o caminho é fechado.

Formalmente um caminho c é dito ser d-separado (ou bloqueado) por um conjunto de nós **Z** se e somente se:

- c contem uma cadeia i $\rightarrow$ m $\rightarrow$ j ou uma divergência i $\leftarrow$ m $\rightarrow$ j tal que o nó do meio m está em **Z**;
- c contem uma convergência (ou colisor) i $\rightarrow$ m $\leftarrow$ j tal que o nó do meio não está em **Z** e nenhum descendente de m está em **Z**

O conjunto **Z** d-separa **X** de **Y** se e somente se, **Z** bloqueia todos os caminhos de nós em **X** para nós em **Y**.

Se um caminho satisfaz a condição acima, diz-se que ele é bloqueado; caso contrário ele é ativado por **Z**. Na figura 3.2 $X = 2$ e $Y = 3$ são d-separados por Z = 1; o caminho $2 \leftarrow 1 \rightarrow 3$ é bloqueado por $1 \in \mathbf{Z}$.

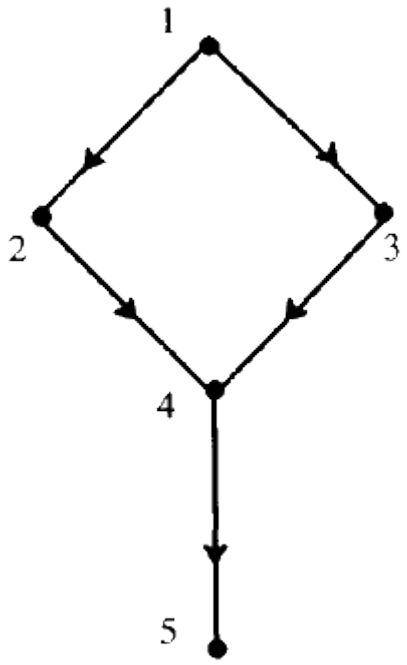

Figura 3.2: Um DAG demostrando d-separation; o nó 1 bloqueia o caminho 2-1-3, enquanto o nó 5 ativa o caminho 2-4-3 (retirado de [1])

Em 1988 Pearl prova que os conjuntos **X** e **Y** são d-separados por **Z** em DAG se e somente se **X** é independente de **Y** condicionado a **Z** em toda distribuição compatível com G.

Em outras palavras $(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z})_G \Leftrightarrow (\mathbf{X} \perp \mathbf{Y}|\mathbf{Z})_P$, onde $(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z})_G$ significa que **X** é d-separado de **Y** dado **Z** em um grafo G, e $(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z})_P$ é a independência estatística como discutido na sessão anterior.

3.3 M-separation

Um teste alternativo para d-separação foi proposto por Lauritzen [11], baseado na noção de grafos ancestrais e foi chamada de m-separação (separação moralizada) por Silva e Ladeira [5].

- Exclua de G todos os nós exceto aquele em anc($\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$);
- Conecte por uma aresta todo par de nós que possuam filho em comum (moralização);
- Remova todas orientações dos arcos

Então $(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z})_G$ se e somente se, $\mathbf{Z}$ intercepta todos os caminhos entre $\mathbf{X}$ e $\mathbf{Y}$ no grafo não orientado resultante. Isto é, se removendo $\mathbf{X}$, $\mathbf{Y}$ e $\mathbf{Z}$ ficam desconectados, então $\mathbf{X}$ e $\mathbf{Y}$são condicionalmente independentes dado $\mathbf{Z}$, caso contrário $\mathbf{X}$ e $\mathbf{Y}$ são dependentes dado Z. Onde (anc(W)) contém os nós de W e todos seus ancestrais.

O Benefício de m-separação sobre a d-separação é o fato de ser um processo algorítmico e portanto fácil de se implementar.

Capítulo 4

Aprendizado de Redes Bayesianas

Muitas vezes, quando queremos construir uma BN, o conhecimento do relacionamento de causa entre as variáveis do nosso dominio pode ser incerto, o custo de um especialista muito elevado, e principalmente, precisar as probabilidades dos de cada nó dados seus pode ser inviável. No entanto, se tivermos dados sobre o problema, isto pode facilitar muito esse processo de criação da BN, tudo que precisamos fazer é adaptar técnicas de aprendizado de máquina para o escopo de BNs. Pensando nisto o interesse em desenvolver e implementar estas técnicas vem aumentando nos últimos anos.

É importante observar que as probabilidades representadas por BN pode ser Bayesiana ou Física. Quando construímos uma BN a partir de conhecimento prévio tão somente, as probabilidades serão Bayesianas. No entanto se aprendermos estas estruturas a partir de dados, estas probabilidades serão físicas [7].

O aprendizado de uma BN é dividido em duas etapas distintas e independentes

- O Aprendizado da Estrutura da rede. Isto é, quais as relações de causas entre as variáveis do nosso domínio.
- O Aprendizado dos Parâmetros da rede. Isto é, dada a estrutura da rede, quais as probabilidades de cada um dos nós.

Obviamente para se aprender os parâmetros de uma rede é necessário que já se tenha a estrutura da pronta. Esta estrutura pode ter sido construída por um especialista, ou aprendida pelos dados. Entretanto o aprendizado de parâmetro é trivial de ser feito e por este motivo já se tornou uma tradição entre as publicações sobre de Aprendizado de Redes Bayesianas apresentar a aprendizagem de parâmetros antes da aprendizagem de estrutura. Nós também seguiremos esta tradição.

4.1 Aprendizado de Parâmetros

O aprendizado de parâmetros nada mais que encontrar a distribuição de probabilidade conjunta de cada variável aleatória presente na reade, representadas por CPTs, dada a topologia da rede.

Seja $\mathbf{X} = X_1, X_2, ..., X_n$ para o conjunto de variáveis aleatórias do model, $B(S)$ para a estrutura da BN e θ_s para os parâmetros da BN. Pearl [1] provou que a função de distribuição conjunta de $\mathbf{X}$ pode ser obitida como o produto das distribuições de probabilidades condicionais da variável da BN, dado os seus pais. A partir da Equação 3.2 temos que:

$$P(\mathbf{X}|\theta_s, B(S)) = \prod_{i=1}^{n} p(x_i|pa_i, \theta_i, B(S)) \tag{4.1}$$

Onde θ_i é o vetor de prâmetros para $P(x_i|pa_i, \theta_i, B(S))$, θ_s é o vetor de parâmetros de $(\theta_1, ..., \theta_n)$. O que desejamos é econtrar os parâmetros θ_s dado um conjunto de treinamento D e a estrutura $B(S)$. Para isto avaliamos a expressão $P(\theta_s|D, B(S))$. É importante notar que D deve ser um conjunto de terinamento completo e é respresentado por $x_1, x_2, ..., x_n$, onde cada x_i representa um caso do conjunto de dados observados. As incertezas são codificadas sobre os parâmetros θ_s por uma variável aleatória Θ_s e a função a priori $P(\theta_s|B(S))$. A função $P(x_i|pa_i, \theta_i, B(S))$ é vista como a função de distribuição local.

A partir de manipulações aritméticas que fogem do escopo deste trabalho chegamos na seguinte equação:

$$P(X_i = x_i^k|Pais = pa_i^j) = \frac{\alpha_{ijk} + N_{ijk}}{\sum_i^{r_i} \alpha_{ijk} + N_{ij}} \tag{4.2}$$

Seja o domínio de X_i denotado por D_{x_i}

- X_i é a i-ésima variável da rede bayesiana;
- $x_i^k \in D_{x_i}$ é a k-ésima instância da variável X_i;
- pa_i^j é a j-ésima instância da variável X_i;
- α_{ijk} é o parâmetro da distribuição de Dirichlet;
- r_i é a cardinalidade dos estados da variável X_i, de D_{x_i};
- N_{ij} é o número total de ocorrências de X_i dados os seus pais; pa_i^j, isto é, $N_{ij} = \sum_{k=1}^{n} N_{ijk}$

Os parâmetros α_{ijk} podem ser substituídos por 1, pois corresponde ao valor esperado da frequência de cada estado, admitindo-se uma distribuição uniforme para os estados de

X_i, visto que, a princípio, não se tem nenhuma informação que permita estimação melhor para essa distribuição de probabilidades.

A seguir daremos um exemplo da aplicação dessa fórmula para o aprendizado da rede Ásia 4.1 proposto por Lauritzen e Spielgelhater [12]. A tabela com os dados foi retirada de [4]

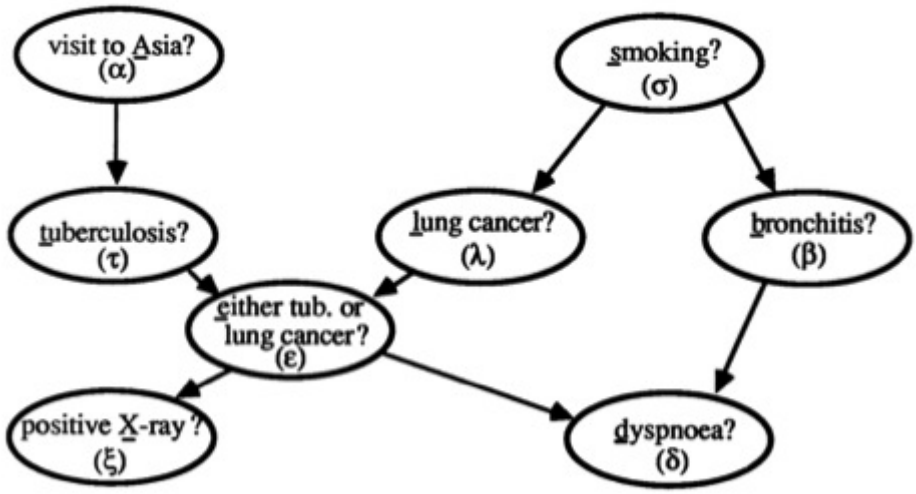

Figura 4.1: Rede Ásia

Exemplo 4.1 *Neste exemplo desejamos aprender os parâmetros para a variável dispnéia da rede Àsia, a partir do conjunto de treinamento apresentado na Tabela 4.2*

Os pais da variável da variável Dispneia são TouC (coluna 4) e Bronquite (coluna 6), portanto $pa_7 = [4, 6]$. Os estados de TouC e Bronquite são, respectivamente representados por:

- $D_x[4] = [0, 1]$, onde 0 representa 'Não' e 1 'Sim';
- $D_x[6] = [0, 1]$, onde 0 representa 'Ausente' e 1 representa 'Presente'.

As possíveis instâncias para Dispneia são: $pa_7 = [[0, 0]^0, [0, 1]^1, [1, 0]^2, [1, 1]^3]$ onde por exemplo a instância 2: $[1, 0]^2$ representa TouC assumindo o valor 'Sim ' e Bronquite assumindo o valor 'Ausente'.

A matriz N_{ijk} para dispnéia, obtida através da contagem na matriz D das ocorrências de Dispneia, condicionadas as instâncias possíveis para os pais TouC e Bronquite, está apresentada na Tabela 4.1

A partir dessa taela é possível calcular a probabilidade associada a cada parâmetro da distribuição de probabilidade da variável aleatória DIspneia. A tabela de probabiliades condicionais de Dispneia, calculada com a 4.2, está representada na Tabela 4.2

	Ásia	Tuberculose	Fumante	Câncer	TouC	RaioX	Bronquite	Dispnéia
01	Não	Ausente	Sim	Ausente	Não	Normal	Presente	Presente
02	Não	Ausente	Sim	Ausente	Não	Normal	Ausente	Ausente
03	Não	Presente	Sim	Ausente	Não	Normal	Ausente	Ausente
04	Não	Ausente	Não	Ausente	Não	Normal	Ausente	Presente
05	Não	Ausente	Sim	Ausente	Não	Normal	Ausente	Ausente
06	Não	Ausente	Sim	Ausente	Não	Normal	Ausente	Ausente
07	Não	Ausente	Não	Ausente	Não	Normal	Presente	Presente
08	Sim	Ausente	Sim	Presente	Sim	Anormal	Presente	Presente
09	Não	Ausente	Não	Ausente	Não	Normal	Ausente	Ausente
10	Não	Ausente	Não	Ausente	Não	Normal	Ausente	Ausente
11	Não	Ausente	Sim	Ausente	Não	Normal	Ausente	Presente
12	Sim	Ausente	Sim	Ausente	Não	Normal	Presente	Presente
13	Não	Presente	Não	Ausente	Não	Normal	Presente	Presente
14	Não	Ausente	Sim	Ausente	Não	Normal	Presente	Ausente

Figura 4.2: Exemplo de conjunto de treinamento para a Rede Ásia

4.2 Aprendizado de Estrutura

O Objetivo da aprendizagem da Estrutura é encontrar a topologia da BN que mais se adéqua aos nossos dados. Para isto podemos atacar o problema de duas formas distintas:

- Busca e pontuação: fazemos uma busca no conjunto de todos DAG existentes entre nossas variáveis usando heurísticas robustas o suficiente.
- Análise de dependência: utilizamos técnicas estatísticas bem desenvolvidas para analisar a dependência entre nossos dados e a partir deles inferir a estrutura da rede.

Vamos começar discutindo sobre algoritmos de busca e pontuação, apresentamos as heurísticas mais famosas.

Tabela 4.1: Matriz N_{ijk} Para Dispneia com Pais T ou C e Bronquite.

	Instancia dos Pais			
Dispneia	0	1	2	3
0	3	2	1	0
1	1	7	0	0

Tabela 4.2: CPT calculada para Dispneia.

	TouC	0	0	0	1
	Bronquite	0	1	0	1
Dispneia	Sim	$\frac{1+3}{2+4} = 0.67$	$\frac{1+2}{2+9} = 0.27$	$\frac{1+1}{2+1} = 0.67$	$\frac{1+0}{2+0} = 0.5$
	Não	$\frac{1+1}{2+4} = 0.33$	$\frac{1+7}{2+9} = 0.73$	$\frac{1+0}{2+1} = 0.33$	$\frac{1+0}{2+0} = 0.5$

4.2.1 Busca e Pontuação

4.2.2 Análise de Dependência

4.3 Aprendizado Incremental

Capítulo 5

UnBBayes

O UnBBayes é a ferramenta que expandiremos ao longo dos próximos semestres para desenvolver novos algoritmos. UnBBayes é uma aplicação open-source feita em Java™ desenvolvido pelo Grupo de Inteligencia Artificial (GIA) do Departamento de Ciência da Computação da UnB no Brasil e provê um framework para construir modelos gráficos probabilísticos e realizar raciocínios plausíveis. Ele apresenta uma Graphical User Interface (GUI), Aplication Program Interface (API) e ainda suporte a plug-ins para extensões não previstas.

[15] descreve os três objetivos principais das mais novas versões do UnBBayes, são eles:

- Ser uma plataforma para a disseminação dos conceitos e utilidade do raciocínio probabilístico;
- Ser uma ferramenta visual fácil de usar e configurar;
- Fornecer extensibilidade.

O Primeiro objetivo é atingido com a implementação do estado da arte de BN e um algoritmo de inferência padrão baseado no algoritmo de Junction Trees. O segundo através de uma a implementação de GUI intuitiva. E a terceira através da natureza orientada a objetos do Java™ em conjunto com um design de plug-ins.

O projeto oferece um repositório que incluem plug-ins para BN, Influence Diagram (ID), Multiple-Sectioned Bayesian Network (MSBN), Multi Entity Bayesian Network (MEBN), Probabilistic Web Ontology Language (PR-OWL), aprendizado de parametros, aprendizado de estrutura e aprendizado incremental de BNs, data mining, avaliação de classificação e perfomance e análise estatística de dados e finalmente diversos algoritmos para inferencia Bayesiana. A Figura 5.1 ilustra um diagrama de caso de uso do UnBBayes core[1].

[1] O core é uma aplicação com um mínimo de funcionalidades.

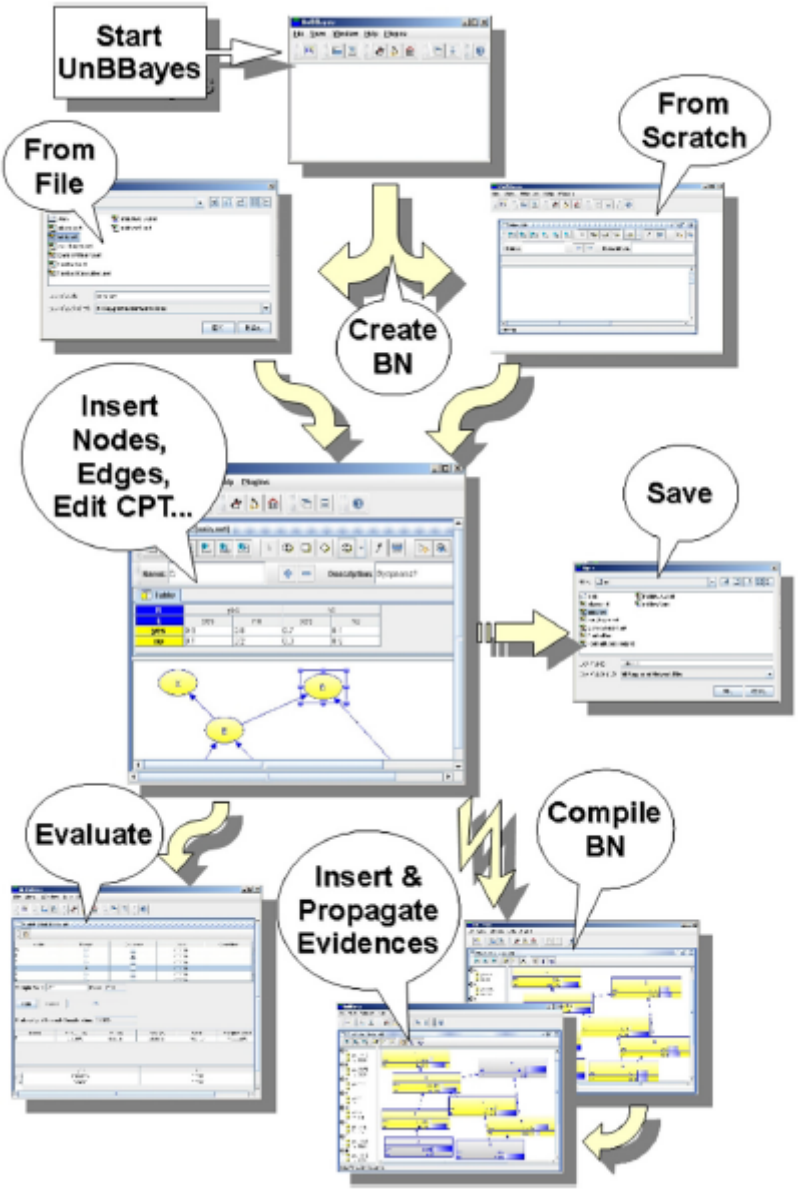

Figura 5.1: Um exemplo de diagrama de uso do UnBBayes core (módulo de BN e da GUI (Retirado de...)

5.1 Design Geral

O UnBBayes é basicamente estruturado no design pattern Model View Controller (MVC) (ver capítulo 6).

A Figura 5.2 ilustra o design geral do UnBBayes.

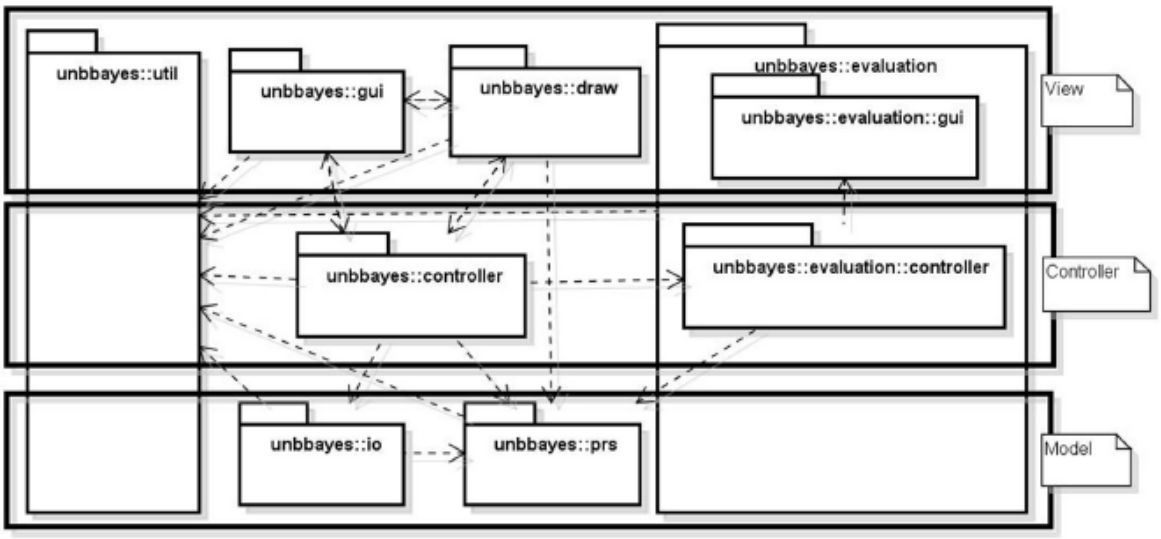

Figura 5.2: uma classe UML ilustrando o design MVC do UnBBayes, em nível de pacotes. (Retirado de [15])

Apesar do UnBBayes ser escrito em Java™ que é uma linguagem tipicamente Orientado a Objetos (OO), o UnBBayes raramente aplica uma orientação a objetos pura em seu design. Mas ao invés, ela segue uma adaptação do paradigma orientado a componentes, o qual organiza os pedaços de código em conjuntos destacáveis e substituíveis. Componentes como num circuito eletrônico, como por exemplo circuitos integrados.

Designs baseados em componentes provêem uma alta variabilidade e reusabilidaide, o que são alguns dos grandes requerimentos mencionados na seção anterior. Isto porque para fazer modificações no programa, basta substituir aquele componente por um outro. Por exemplo se precisarmos trocar o formato de um nó basta substituir o objeto que o rederiza. O que significa dizer que não há necessidade de entrar dentro de uma classe já pronta e configurar aquelas linhas de código.

O módulo core do UnBBayes provê duas funcionalidades mestras: a infraestrutura base (por exemplo, suporte às telas, acesso aos arquivos, suporte a plug-ins) e uma manipulação de BN com funcionalidades básicas (por exemplo inferência e manipulação). A seguir listamos os pacotes que compõem o core (a Figura 5.2 apresentou suas dependências gerais).

- unbbayes.controller: Contem as classes do controller do MVC
- unbbayes.draw: Contem os adapters para determinar como um grafo é desenhado na tela. Permite diversas funcionalidades para desenhar, incluindo redimencionar, colorir, mover, etc.
- unbbayes.evaluation: Contém classes para estimar a BN

- unbbayes.gui: contem as Classes das GUI (por exemplo panels, windows, forms, tables).
- unbbayes.io: contem classes para armazenamento de BN, configurações, preferencias, logs como arquivos.
- unbbayes.prs: contem classes representando estrutura de dados e algoritmos (classes da model do MVC.) Elementos conceituais (Node, Edge, Network, JunctionTree, PotentialTable) e as operações entre eles . PRS vem de Probabilistic Reasoning System, isto é, o sistema de raciocinio probabilistico. Classes nesse pacote representam a BN, classes no unbbayes.prs.id representa uma ID, e classes no unbbayes.prs.hubrdbn representa uma hybrid BN, que inclui nós contínuos com valor númerico.
- unbbayes.example: contem classes exemplificando o uso da API do UnBBayes.
- unbbayes.util: este pacote unifica as classes de utilidade, os quais são classes de propósito gerais utilizados. Classes colocadas aqui provêm as seguintes funcionalidades: debugging, classes abstratas para design patterns e manipulação de coleções avançadas.
- *.resources: pacotes com este padrão de nome contêm classes de recursos, os quais provêem funcionalidade de lcalização. O class loader32 do UnBBayes pode selecionar as classes de recursos apropriados automaticamente, dependendo das configurações do computador.
- *.extension: pacotes com esse padrão de nome contêm classes implementando funcionalidades relacionadas à infra-estrutura de plug-in.
- *.exception: pacotes com esse padrão de nome contem classes representando classes de excessão.

5.1.1 Classes da View e Controller

A lista a seguir oferece uma breve descrição das classes ilustradas na Figura 5.3

- NamedWindowBuilder: este é um builder que instancia uma janela interna. Cada janela instanciada por estes builders terão um identificador (nome) e será colocado no MDIDesktopPane. Esta classe elimina dependencia direta do MainController às classes do módulo de BN (por exemplo NetworkWindow)
- NetworkWindowBuilder: a builder para NetworkWindow

- MainController: este é o controlador responsável por criar, carregar e salvar redes que o UnBBayes suporta. Também lida com as configurações de sistema.
- UnBBayesFrame: esta é uma extensão do swing JFrame responsável pelo painel principal do UnBBayes.
- MDIDesktopPane: Esta é uma extensão do swing JDesktopPane e oferece suporte à funcionalidade Multiple Document Interface (MDI) (isto é, multiplas janelas dentro de uma janela pai). Esta classe também lidai com as barras de scroll para quando a janela interna move muito para a direita ou para baixo da janela principal.
- NetworkController: este é o controlador responsável por engatilhar as operações do módulo de BN (tais como inserção de nós e propagação de evidências).
- NetworkWindow: esta é o frame interno onde a edição de BN é feita.
- PNCompilationPane: este é o painel da NetworkWindow responsável por mostrar a BN compilada e oferecer os meios para inserir e propagar evidências.
- GraphPane: este é o painel da NetworkWindow responsável por mostrar a representação gráfica da BN sendo editada.
- PNEditionPane: este é um painel na NetworkWindow responsável por prover as ferramentas para edição da BN
- UShape: esta classe é responsável por renderizar os nós em um canvas. Encapsulando a classe da model Node.
- UCanvas: esta classe representa o canvas onde o grafo é pintado.
- UShapeLine: esta classe é responsável por renderaziar os arcos no canvas. Ecapsulando a classe da model Edge.

5.1.2 Classes da Model

Classes desta categoria representam dados dentro da memoria. A lista a seguir oferece uma breve descrição das classes da Figura 5.4:

- Graph: common interface for a graph built on a set of nodes and edges
- Network: concrete implementation of a generic network. If a network is composed of probabilistic nodes, using a ProbabilisticNetwork (an extension of Network) would be useful.

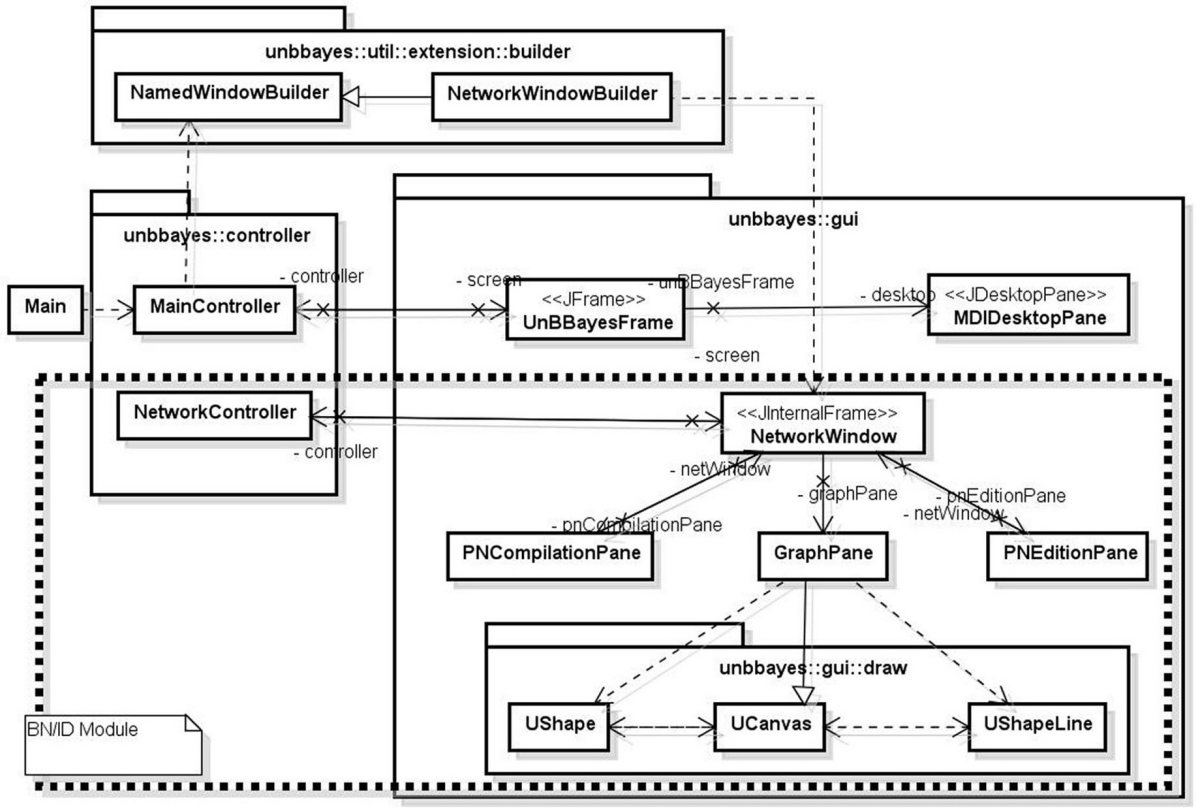

Figura 5.3: Classes do UnBBayes (core) implementando a View e o Controller do MVC (Retirado de [15])

- Edge: this is a class representing an edge between two nodes. By modeling relationships as a separate class, it becomes possible to use attributes, thus allowing differential treatment by other classes (e.g. how the GUI should display a relationship between two nodes depending on the edge's attributes). Technically, it is possible to create relationships (dependencies) between instances of Node without using instances of Edge, but in such case no arcs linking these two nodes will be rendered in the canvas. By doing so, it becomes possible to model "hidden" relationships.
- INode: interface representing a generic node.
- Node: abstract class representing a node. It contains additional information, such as: name, label, description, coordinates, flags36.
- TreeVariable: this is an abstract class for variables that will be displayed in the left side's swing JTree in the compilation panel. After compiling a BN, nodes that are not extending this class will be partially ignored in PNCompilationPane.

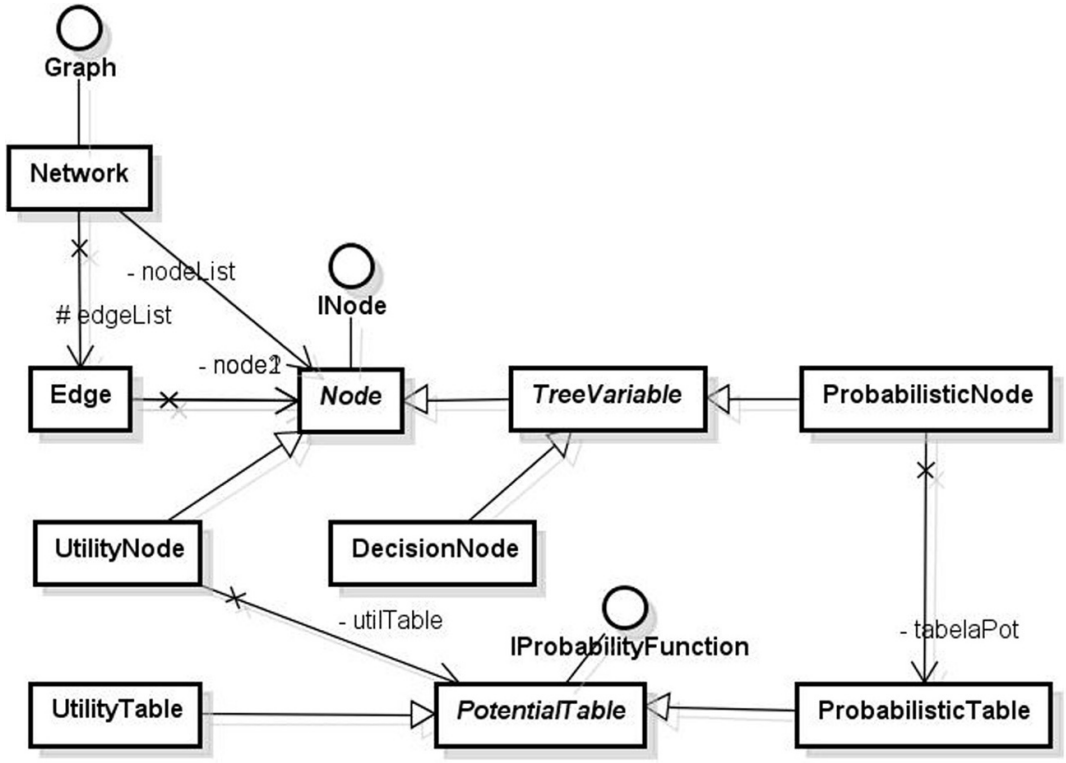

Figura 5.4: Classes do UnBBayes (core) implementando o Model (tipo de dados e operações) do MVC (Retirado de [15])

- ProbabilisticNode: it represents a probabilistic node (i.e. a node having a probabilistic assignment of values).
- UtilityNode: this is an ID's utility node.
- UtilityTable: this is a table representing an ID's utility function (which is represented as a table in UnBBayes' model).
- DecisionNode: this is an ID's decision node.
- IProbabilityFunction: this is a common interface for objects specifying a node's probabilistic distribution.
- PotentialTable: this is an abstract class representing IProbabilityFunction in a table-like format.
- ProbabilisticTable: this class represents a BN's CPT. Objects (instances) of unbbayes.gui.table.GUIPotentialTable can be used to render objects of this class graphically

5.2 Plugins

Java^{TM} oferece um mecanismo de fácil reuso de classes. Incorporando arquivos bytecode Java (".class"ou ".jar") no class path de outros pogramas, classes públicas e métodos serão disponibilizado ao programa como uma biblioteca. Como UnBBayes é disponibilizado como um arquivo ".jar", adicioná-lo ao class path de outro programa irá garantir acesso a sua API.

Diferentemente de API's, plug-ins oferecem meios de rodar um novo código dentro do ambiente de execução do UnBBayes. Um plug-in é um programa que interage com a aplicação hospedeira (um core) e prover dada função. A ligação entre o plug-in e a aplicação core costuma acontecer em tempo de carregamento (quando a aplicação inicializa) ou em tempo de execução. Portanto, designs orientados a plugins oferecem ambientes práticos e flexíveis para prover variabilidade de software, uma vez que nenhuma modificação na aplicação hospedeira é necessária.

5.2.1 Como desenvolver plugins

Capítulo 6

Qualidade de Código e Design Patterns

Quando falamos de projetos de grande e médio porte, a manutenibilidade e fácil adição de novas funcionalidades são preocupações essenciais. Pensando nisso, um grupo de engenheiros de software experientes (muitas vezes denominados Gang of Four (GOF)) se juntaram para compartilhar algumas das soluções dos problemas que eles encontraram ser os mais recorrentes no mundo da engenharia de software e gerencia de projetos no livro *Design Patterns: Elements of Reusable Object-oriented Software*[6]

Segundo o GOF Design Patterns são soluções de padrões de problemas que ocorrem de repetidamente nos projetos orientado a objetos. A ideia é oferecer soluções de modelagens aos projetos em nível abstrato, tirando o máximo proveito de polimorfismo, de herança e da estruturação de classes em geral.

A aplicação de design patterns é importante por diversos motivos, entre eles: Facilitam a comunicação entre desenvolvedores, aumenta a possibilidade de reuso (diminuindo, assim, o tempo de desenvolvimento e aumentando a produtividade), melhora a escalabilidade e qualidade, amplia a visão do desenvolvedor forçado-o a pensar em diversos níveis de abstração e finalmente torna o programa fácil de ser usado por terceiros.

Pelo fato de o UnBBayes ser um projeto grande o suficiente, e que tem como meta oferecer extensiblidade, é de suma importância projetar uma boa estrutura de código e manter o "código bonito", por assim dizer. Para isto, foi dado a nós, que estamos iniciando como desenvolvedores do projeto, um rápido treinamento sobre Design Patterns e Qualidade de Código.

Neste capítulo explicaremos conceitos importantes para o desenvolvimento orientado a objeto, oferecemos uma rápida explicação sobre os três tipos principais de Design Pattern, e os principais exemplos de cada um desses três tipos. No Anexo 1 oferecemos as figuras dos padrões UML de cada um dos Design Pattern mencionados

6.1 Conceitos Preliminares

- Acoplamento: O quanto duas (ou mais) classes dependem uma da outra para funcionar. Um alto acoplamento diminui a manutenibilidade do código pois as alterações em um código poderá ter grande impacto em todo o projeto. O ideal é que as alterações sejam atômicas, isto é, quanto menos uma alteração afetar o programa como um todo, melhor. Podemos dizer sem perda de generalidade que a essência de uma boa OO é reduzir o acoplamento.
- Coesão: Funcionalidades "bem empacotadas", isto é, cada classe, pacotes ou bibliotecas tem sua funcionalidade muito bem definida e especificada. Seus artefatos representam uma mesma categoria lógica.
- Binding: Dependências entre classes e objetos
- Open-Closed: Aberto para extensão, Fechado para alteração de fonte.
- Inversão de Controle: Separar "o que"de "quando".
- Injeção de Dependência: Especifica o objeto com o qual deve ser usado para resolver algum problema. É uma das formas de realizar inversão de Controle.
- Orientação a componentes: componentes reutilizáveis são artefatos auto-contidos, claramente identificáveis, que descrevem ou realizam uma função específica e têm interfaces em conformidade com um dado modelo de arquitetura de software, possuindo documentação apropriada e um grau de reutilização definido[17].

A ideia chave de Design Patterns é aumentar a qualidade do nosso código. Para isto, podemos seguir algumas regras básicas, tais como:

- Evitar if-else, switch-case sempre que possível utilizar polimorfismo e injeção de dependências.
- Usar interfaces em assinaturas de métodos ao invés de classes. Desta forma faremos herança única, implementação múltipla.
- Uso de protected para reforçar o conceito de open-closed.
- Métodos privates para código muito especifico ou quando afeta na segurança.
- Sempre usar private em atributos.

6.2 Padrões de Construção

Padrões de construção abstraem o processo de instanciação. Eles podem ajudar a fazer o sistema independente de como seus objetos são criados, compostos e representados. Uma classe do padrão de construção usa herança para variar a classe que está sendo instanciada, enquanto o objeto do padrão erá delegar a instanciação para outro objeto.

Alguns dos padrões de construção mais importantes são:

- Singleton: Garante somente uma instanciação do objeto no sistema. (ver Figura I.1)
- Factory Method: Criação de objeto sem expor a lógica de criação para o cliente. (ver Figura I.2)
- Builder: Separa a classe do seu construtor. É útil quando o construtor é complexo demais.(ver Figura I.3)
- Prototype: Clona objetos(ver Figura I.4)

6.3 Padrões de Estrutura

Padrões de estrutura estão preocupados em como classes e objetos são compostos para formarem estruturas maiores. Elas usam herança para compor interfaces ou implementações.

Alguns dos padrões de estrutura mais importantes são:

- Adapter: Provê a ponte entre duas interfaces incompatíveis.(ver Figura I.5)
- Bridge: Provê uma interface que faz as funcionalidades de classes concretas trabalhem juntas de tal forma que alterar uma destas classes concretas não altera a outra.(ver Figura I.6)
- Composite: Cria uma classe que contém grupos de de seu próprio objeto.(ver Figura I.7)
- Decorator: Cria uma classe decoradora que encapsula a classe original e provê funcionalidades adicionais, mantendo as assinaturas dos métodos originais intactas.(ver Figura I.8)
- Facade: Cria uma única classe que provê métodos simplificados requeridos pelo cliente e delega as chamadas para classes do sistema existente.(ver Figura I.9)
- Proxy: Cria um objeto proxy que possúi o objeto original para fazer a interface de suas funcionalidades para o "mundo externo".(ver Figura I.10)

6.4 Padrões de Comportamento

Os padrões de comportamento se preocupam com os algoritmos e a distribuição de responsabilidades entre os objetos. Eles não são apenas padrões entre objetos e classes, mas também padrões de comunicação entre eles. Estes padrões caracterizam fluxo de controle difíceis de seguir em tempo de execução. Eles afastam o foco do fluxo de controle para permitir que se concentre somente na maneira como os objetos se comunicam entre si.

Alguns dos padrões de estrutura mais importantes são:

- Chain of Responsibility: Este padrão desacopla senders e receivers do pedido baseado no tipo de pedido. Normalmente cada cada recebedor contem referencia pra outro recebedor. Se um dos objetos não souber lidar com o pedido então ele repassa o pedido para o próximo receiver e assim por diante. (ver Figura I.11)
- Command: Um pedido é empacotado por um objeto como um comando e passado para o objeto invocador. O invocador procura o objeto apropriado para lidar com este comando.(ver Figura I.12)
- Interpreter: Provê uma forma de avaliar uma linguagem gramatical ou expressão.(ver Figura I.13)
- Iterator: Provê meios de acessar elementos de uma coleção de maneira sequencial.(ver Figura I.14)
- Mediator: Provê uma classe que lida com todas comunicações entre diferentes classes.(ver Figura I.15)
- Null Object: Provê uma classe que toma o lugar do objeto null. Muito utilizado para implementar uma "do nothing relationship".(ver Figura I.16)
- Observer: Utilizado em relações um-para-muitos entre interdependentes objetos de tal forma que se um deles mudar o outro é notificado automaticamente.(ver Figura I.17)
- State: Cria um objeto que representa vários estados e um objeto de contexto cujo comportamento varia de de acordo com a mudança do objeto de estados.(ver Figura I.18)
- Template: É uma classe abstrata que define uma forma padrão de executar dado método. Isto é, as subclasses não podem fazer override desta implementação.(ver Figura I.19)

- Visitor: Provê meios de mudar o algoritmo de execução de uma dada classe. Desta forma, o algorimo de execução pode variar de acordo com as variações do visitor.(ver Figura I.20)

Capítulo 7

Resumo das Atividades

Durante esse semestre de 2015/2 foram iniciados os estudos em conjunto com o Guilherme Carvalho para que desenvolvamos nosso TCC (independentemente) nos semestres que estão por vir.

Este capítulo tem por objetivo oferecer um resumo das atividades desenvolvidas no decorrer deste semestre e delinear as atividades a serem feitas nos semestres seguintes.

7.1 O que foi feito

No decorrer deste semestre o professor Shou Matsumoto, douturando da GMU ministrou aulas semanais às segundas via Skype sobre os conteúdos de:

1. Preparação do ambiente UnBBayes. Com o objetivo de instalar o Maven e o Subversion no eclipse e deixar o ambiente pronto para trabalhar, consertando todos possíveis problemas de dependencias;
2. Design Pattern e Anti-Patterns;
3. Arquitetura do UnBBayes;
4. Como desenvolver um Plug-in para o UnBBayes;
5. Redes Bayesianas;
6. MEBN;
7. PR-OWL;
8. Diagramas de Influência.

No decorrer do semestre também fui escalado para implementar ou consertar tarefas de usabilidade no UnBBayes. Foram elas:

1. Implementar crtl+c e crtl+v de nós para a extensão MEBN;
2. Implementar deleção de multiplos findings de MEBN;
3. Refatorar a CPT para prover uma melhor visualização de tabelas muito grandes.

Para a primeira tarefa tentei refatorar o crtl+c crtl+v já existente no core para que cada nó saiba como se copiar, entretanto tive problemas com as chamadas de funções, pois a funcionalidade de cada nó não estava encapsulada o suficiente para fazê-lo, então por motivo de falta de tempo não pude ir adiante.

A terceira tarefa foi implementada através da não-scrollagem da primeira coluna.

7.2 O que será feito

Ficou decidido que nos semestres subsequentes será implementado um novo módulo de Incremental Learning, utilizando-se as mais novas tecnicas disponíveis. E com isto analisar dados de Bactérias disponibilizados pelo professor Ladeira.

Uma idealização do cronograma a ser realizado nos próximos semestres pode ser encontrado em [14]

Capítulo 8

Conclusão

Referências

[1] Judea Pearl (Auth.). *Probabilistic Reasoning in Intelligent Systems. Networks of Plausible Inference*. Elsevier Inc, 1988. 8, 11

[2] Thomas Bayes. An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, f. r. s. communicated by mr. price, in a letter to john canton, m. a. and f. r. s. *Philosophical Transactions of the Royal Society*, 53:370–418, 1763. 4

[3] Jie Cheng, Russell Greiner, Jonathan Kelly, David Bell, e Weiru Liu. Learning bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137(1–2):43 – 90, 2002. 7

[4] Danilo Custodio da Silva. Aprendizagem estrutural de redes bayesianas com dados massivos, 2005. 12

[5] M. Ladeira; W. da Silva. Mineração de dados em redes bayesianas. *Anais do XXII Congresso Brasileiro de computação.: Florianópolis: SBC, 2002*, pages 235–286, 2002. 8

[6] Erich Gamma, Richard Helm, Ralph Johnson, e John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995. 23

[7] David Heckerman. A tutorial on learning with bayesian networks. 156:33–82, 2008. 1, 10

[8] A. N. Kolmogorov. *Grundbegriffe der Wahrscheinlichkeitsrechnung*. Springer, Berlin, 1933. 5

[9] Kevin B. Korb e Ann E. Nicholson. *Bayesian artificial intelligence*. Chapman & Hall / CRC computer science and data analysis. Chapman & Hall/CRC, Boca Raton, Fla., London, 2004. 4

[10] P.S. Laplace e A.I. Dale. *Pierre-Simon Laplace Philosophical Essay on Probabilities: Translated from the fifth French edition of 1825 With Notes by the Translator*. Sources in the History of Mathematics and Physical Sciences. Springer New York, 2012. 4

[11] Steffen L Lauritzen, A Philip Dawid, Birgitte N Larsen, e H-G Leimer. Independence properties of directed markov fields. *Networks*, 20(5):491–505, 1990. 8

[12] Steffen L Lauritzen e David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 157–224, 1988. 12

[13] David Lewis. A subjectivist's guide to objective chance. In Richard C. Jeffrey, editor, *Studies in Inductive Logic and Probability*, volume 2, pages 83–132. University of California Press, 1980. 4

[14] Abreu P. Cronograma para atividades seguintes. `https://docs.google.com/spreadsheets/d/1zkRXaUf4M9W3tRFdGG9CyN5mmGbuR_CcFPDN19t8yTk/edit#gid=2133875924`. 29

[15] Matsumoto, S., Carvalho, R., Ladeira, M., Costa, P., Santos, L., Silva, D., Onishi, e M. & Machado. *UnBBayes: a Java Framework for Probabilistic Models in AI. Java in Academia and Research*, chapter UnBBayes: a Java Framework for Probabilistic Models. 2011. 15, 17, 20, 21

[16] John Venn. *Logic of Chance*. 1866. 4

[17] Cláudia Maria Lima Werner e Regina Maria Maciel Braga. Desenvolvimento baseado em componentes. *XIV Simpósio Brasileiro de Engenharia de Software. João Pessoa, Brasil*, 2000. 24

Anexo I

Figuras dos Design Patterns Mencionados

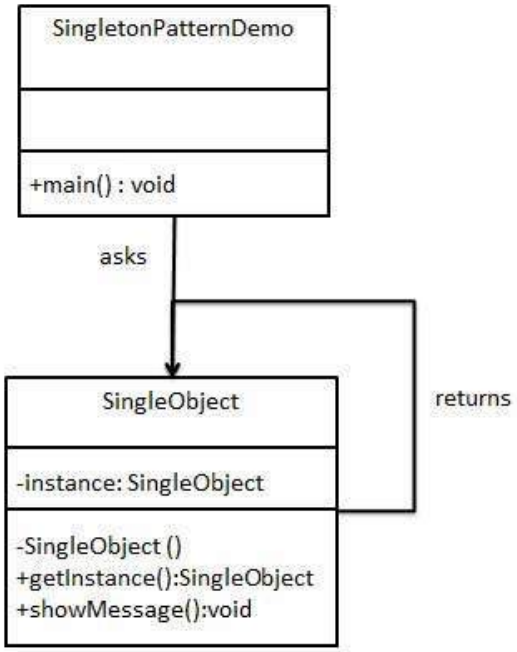

Figura I.1: Diagrama UML do Design Pattern Singleton

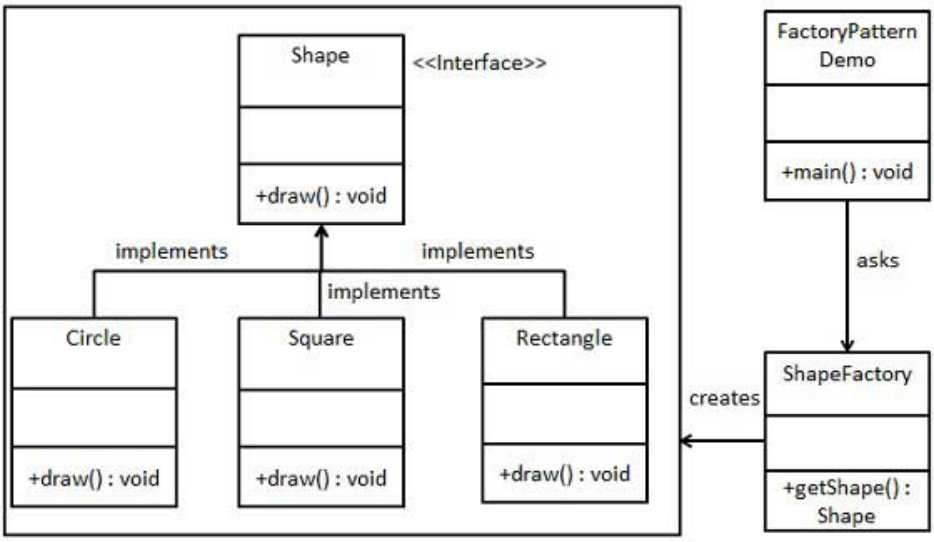

Figura I.2: Diagrama UML do Design Pattern Factory

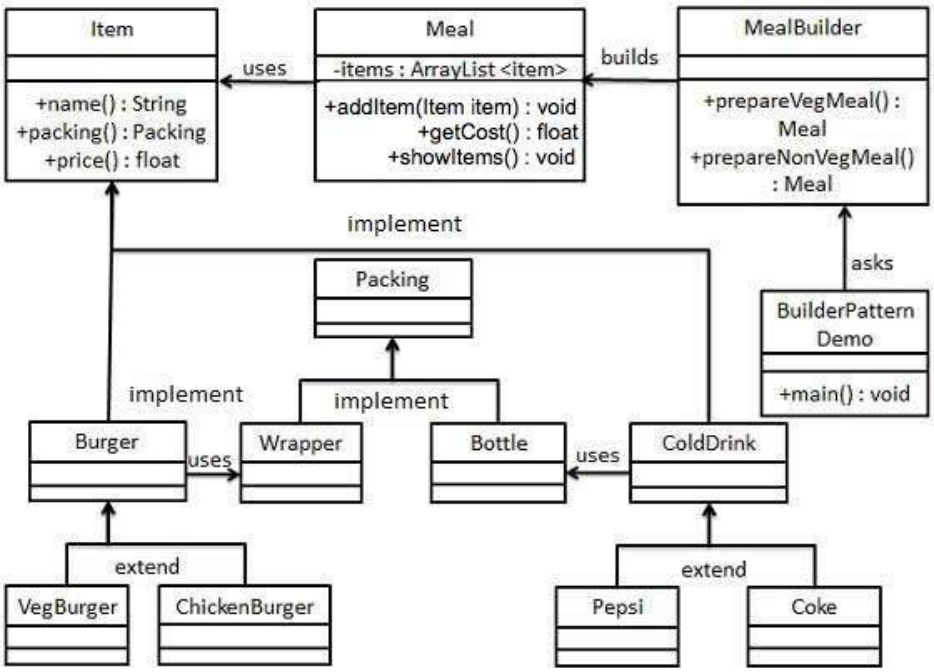

Figura I.3: Diagrama UML do Design Pattern Builder

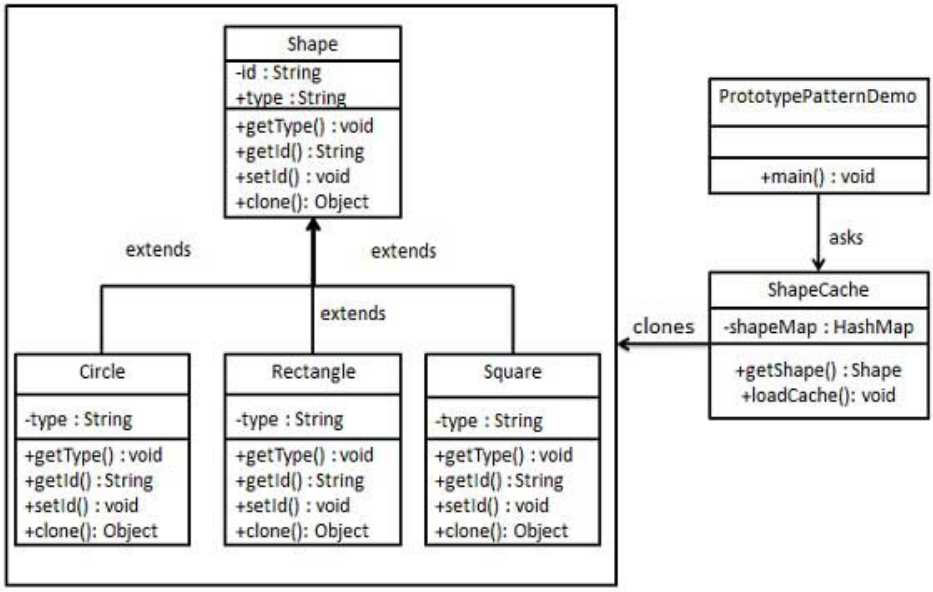

Figura I.4: Diagrama UML do Design Pattern Prototype

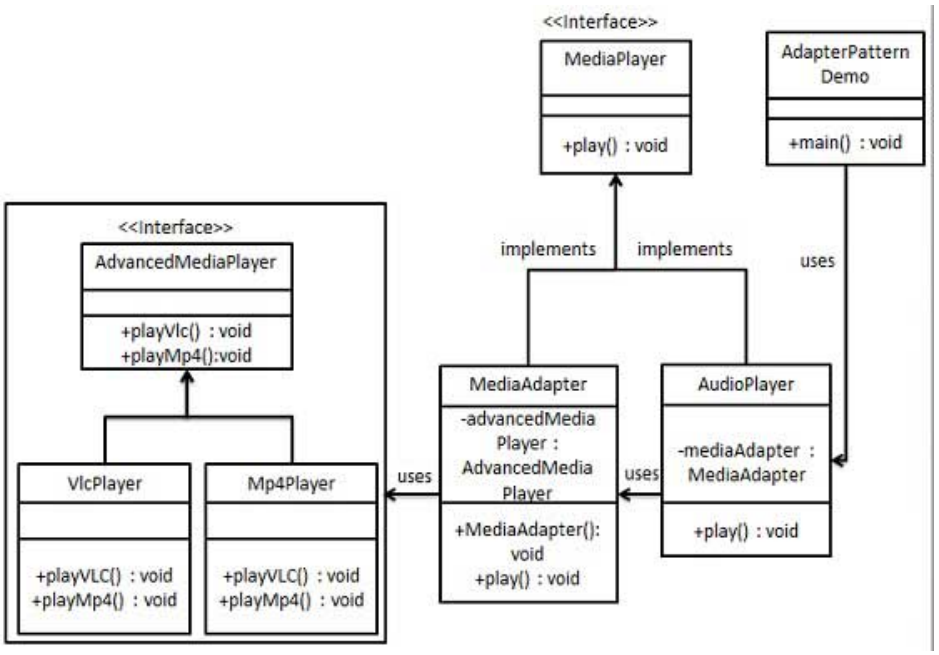

Figura I.5: Diagrama UML do Design Pattern Adapter

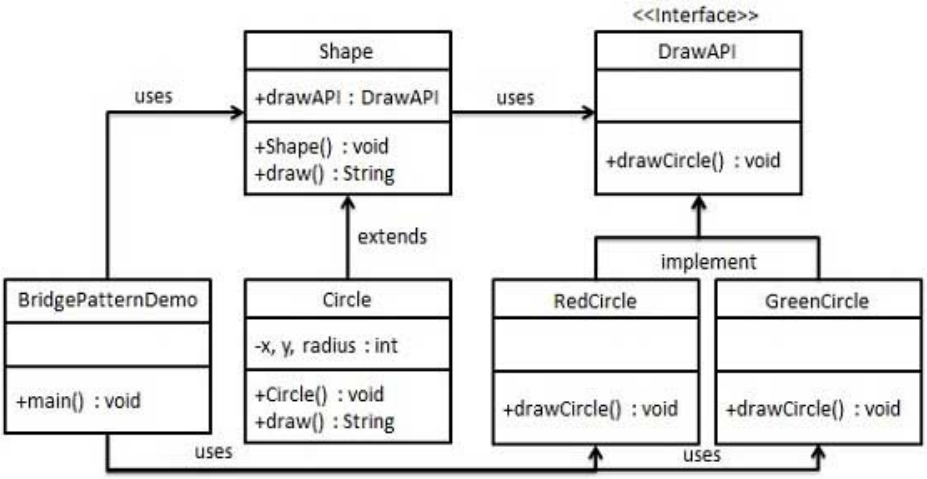

Figura I.6: Diagrama UML do Design Pattern Bridge

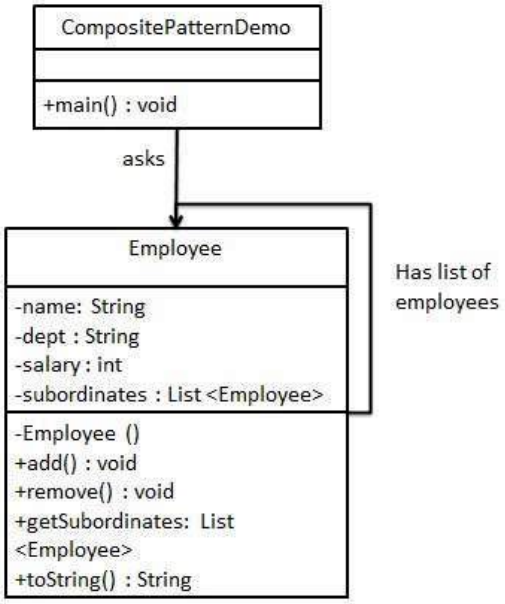

Figura I.7: Diagrama UML do Design Pattern Composite

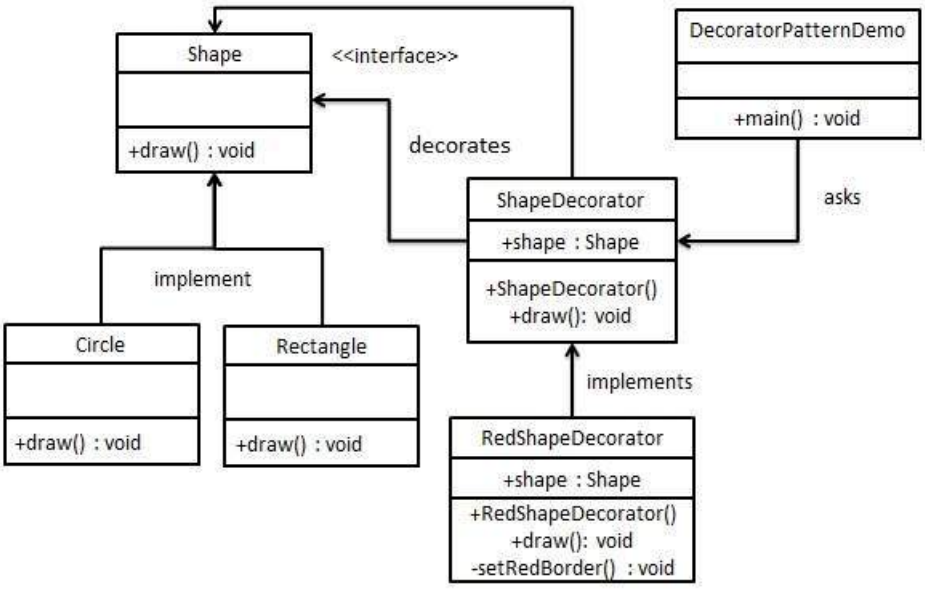

Figura I.8: Diagrama UML do Design Pattern Decorator

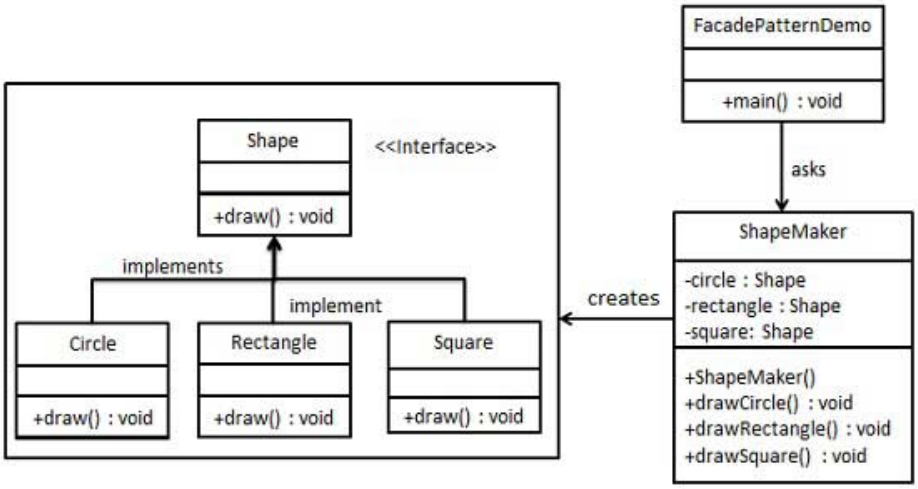

Figura I.9: Diagrama UML do Design Pattern Facade

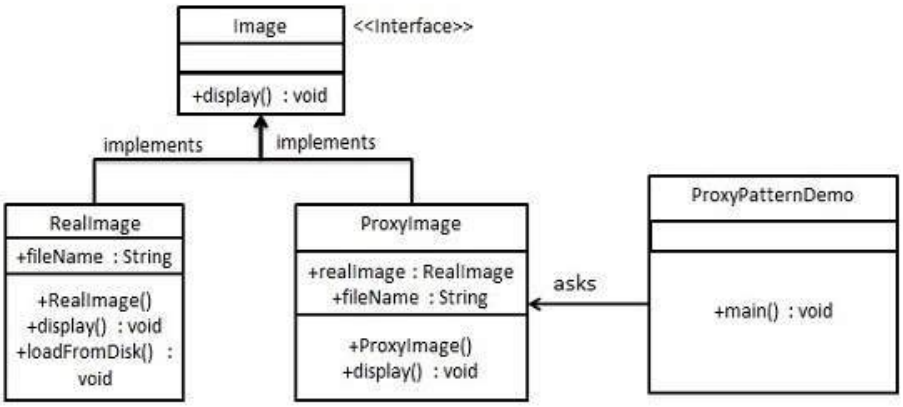

Figura I.10: Diagrama UML do Design Pattern Proxy

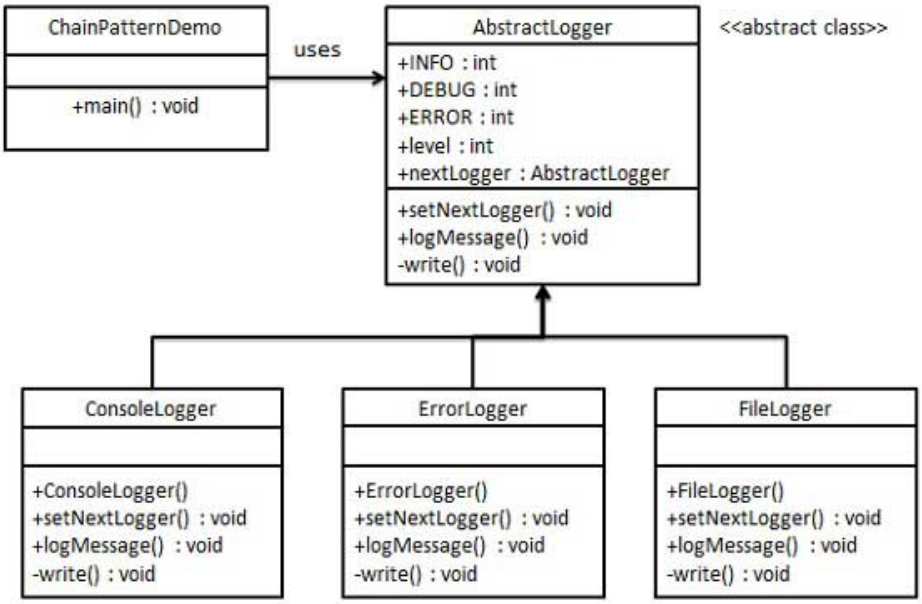

Figura I.11: Diagrama UML do Design Pattern Responsability

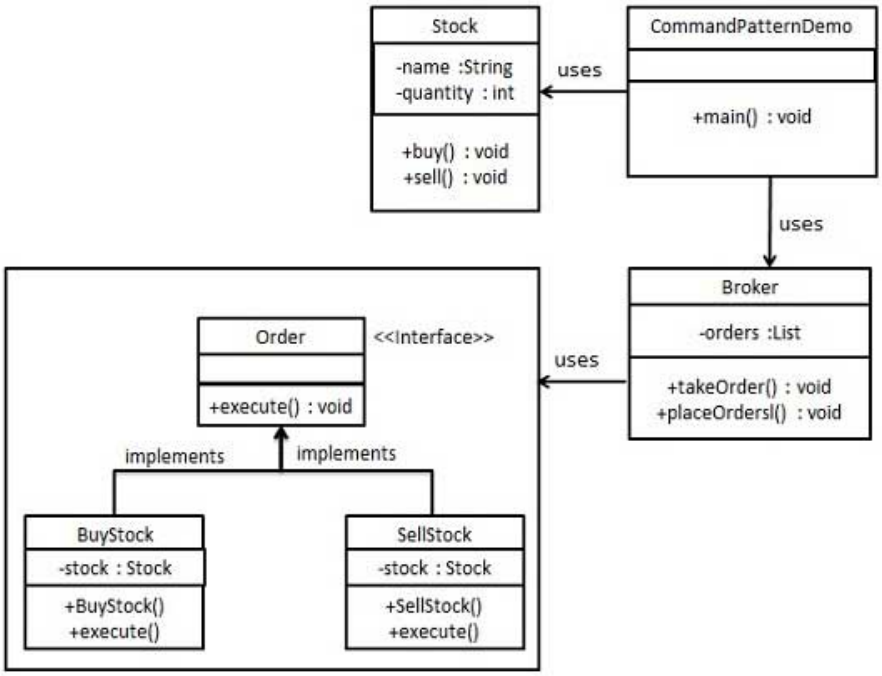

Figura I.12: Diagrama UML do Design Pattern Command

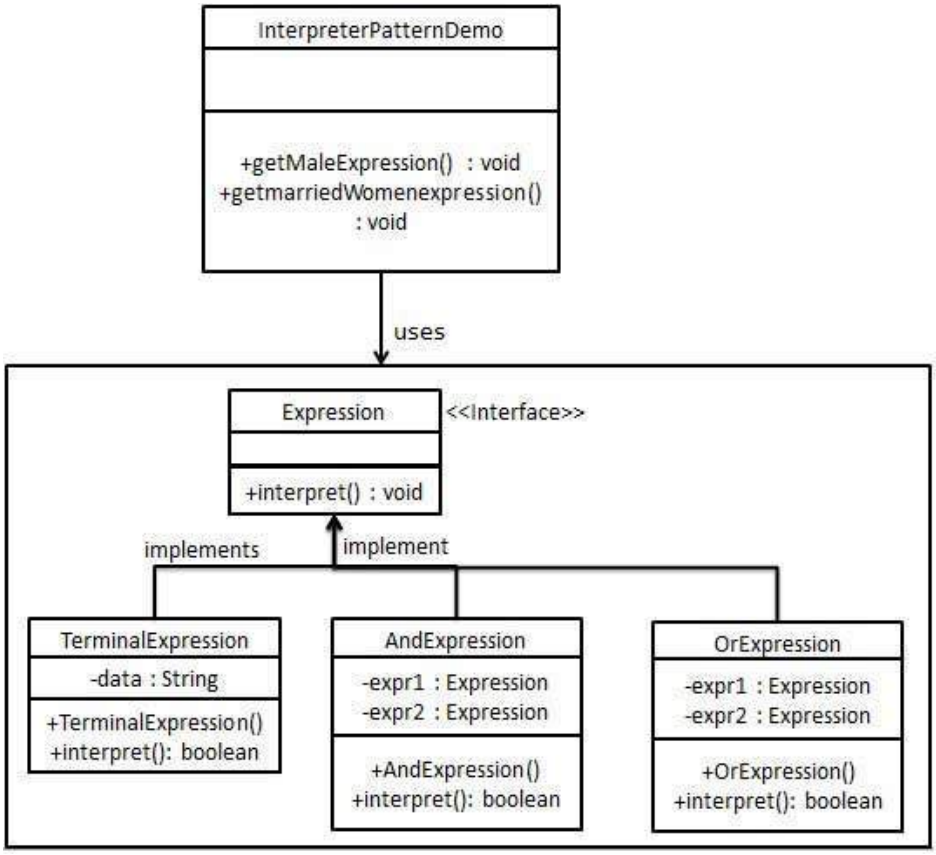

Figura I.13: Diagrama UML do Design Pattern Interpreter

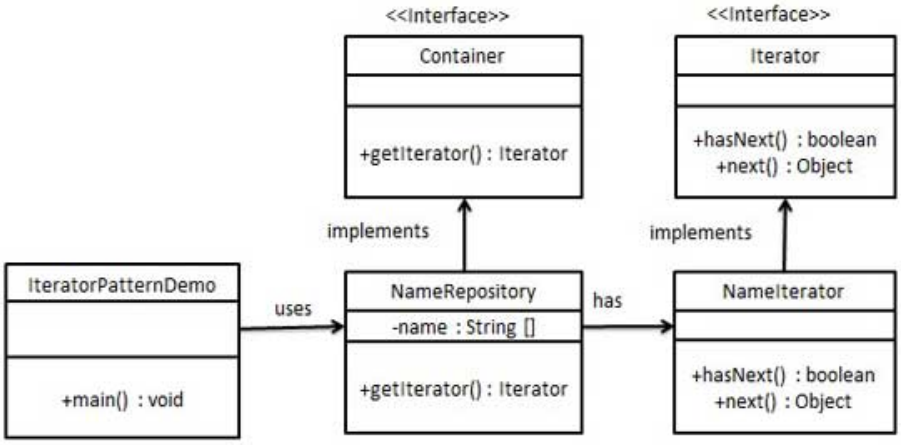

Figura I.14: Diagrama UML do Design Pattern Iterator

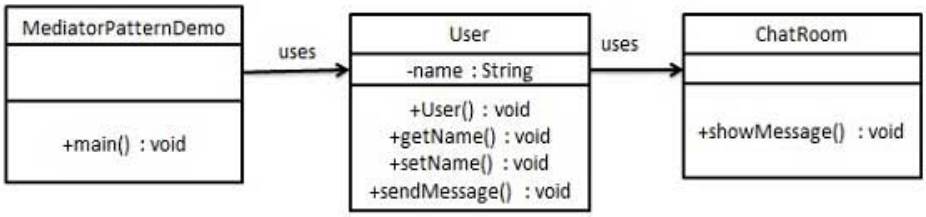

Figura I.15: Diagrama UML do Design Pattern Mediator

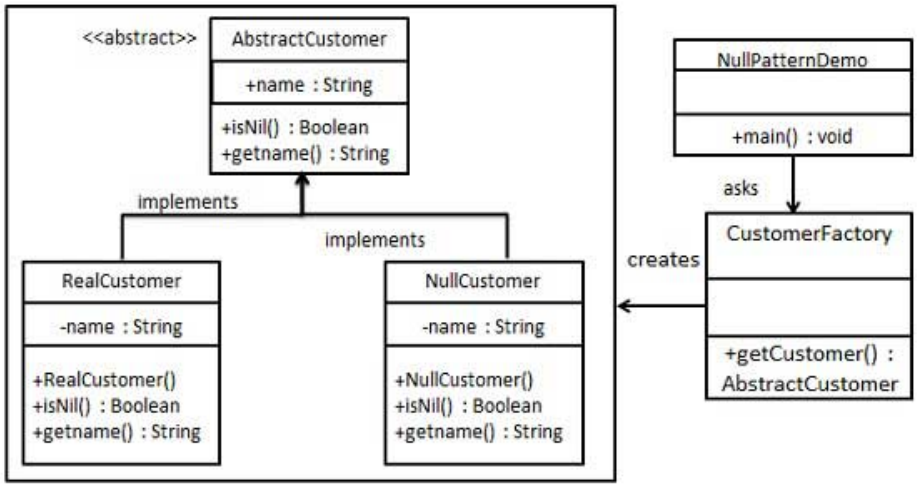

Figura I.16: Diagrama UML do Design Pattern Null Object

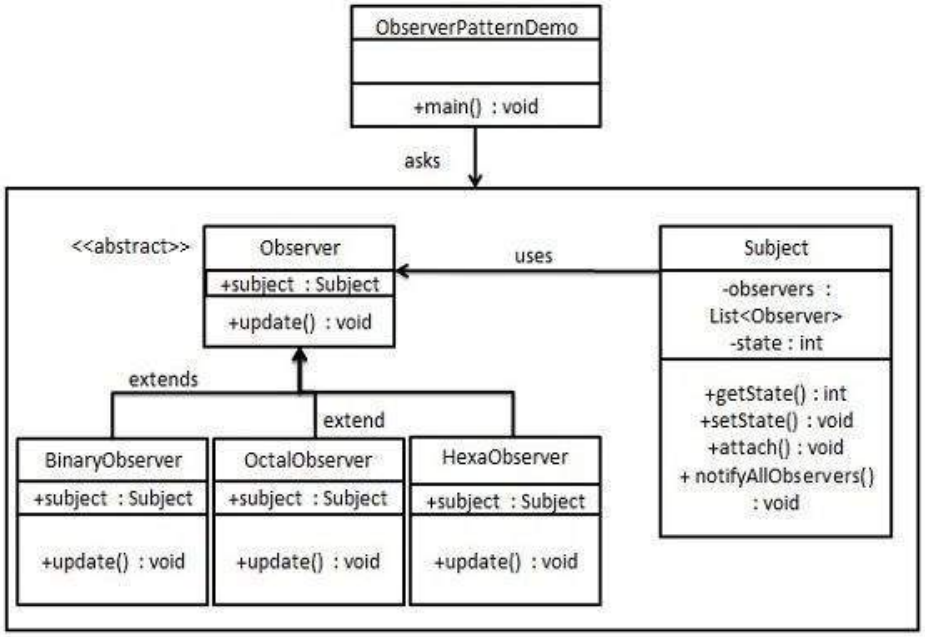

Figura I.17: Diagrama UML do Design Pattern Observer

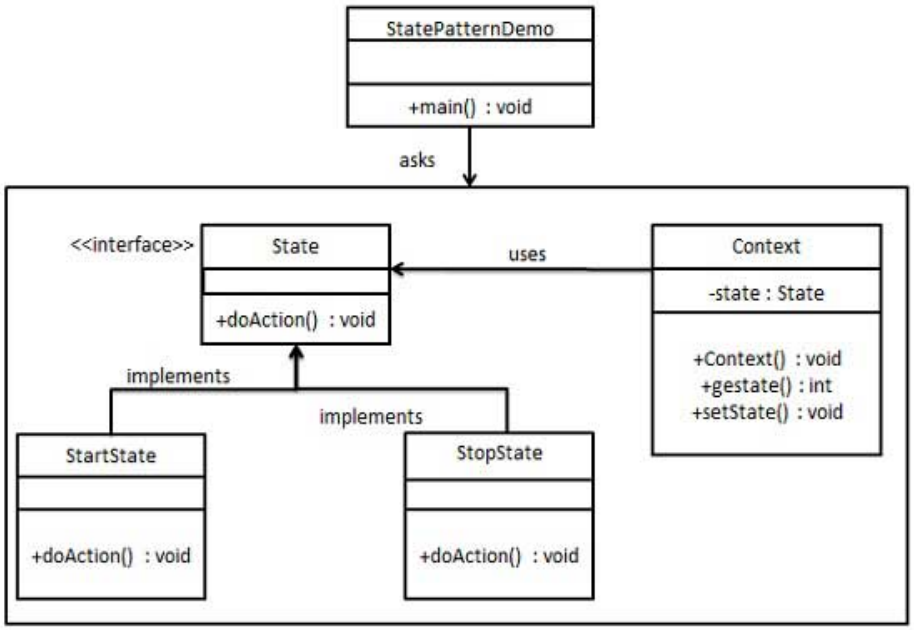

Figura I.18: Diagrama UML do Design Pattern State

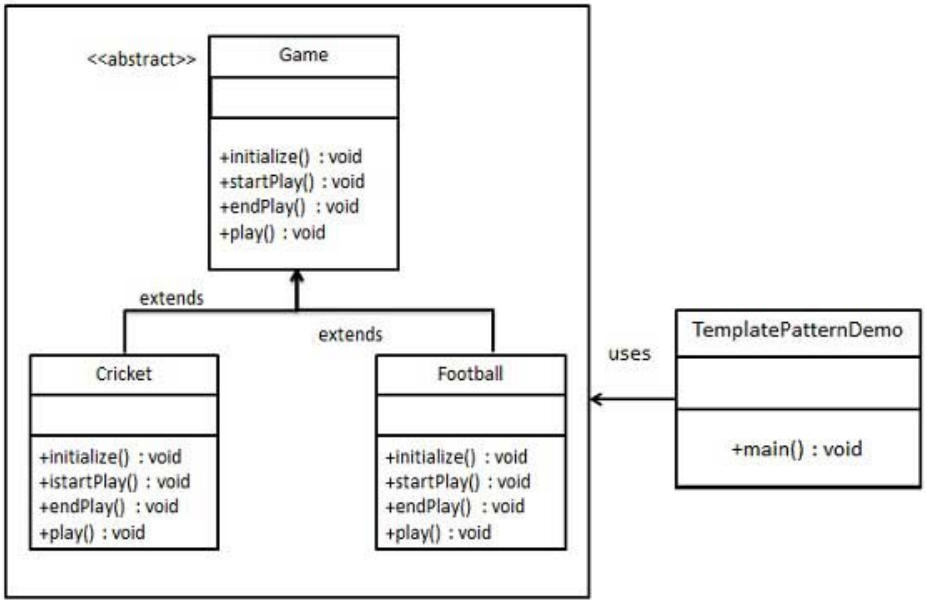

Figura I.19: Diagrama UML do Design Pattern Template

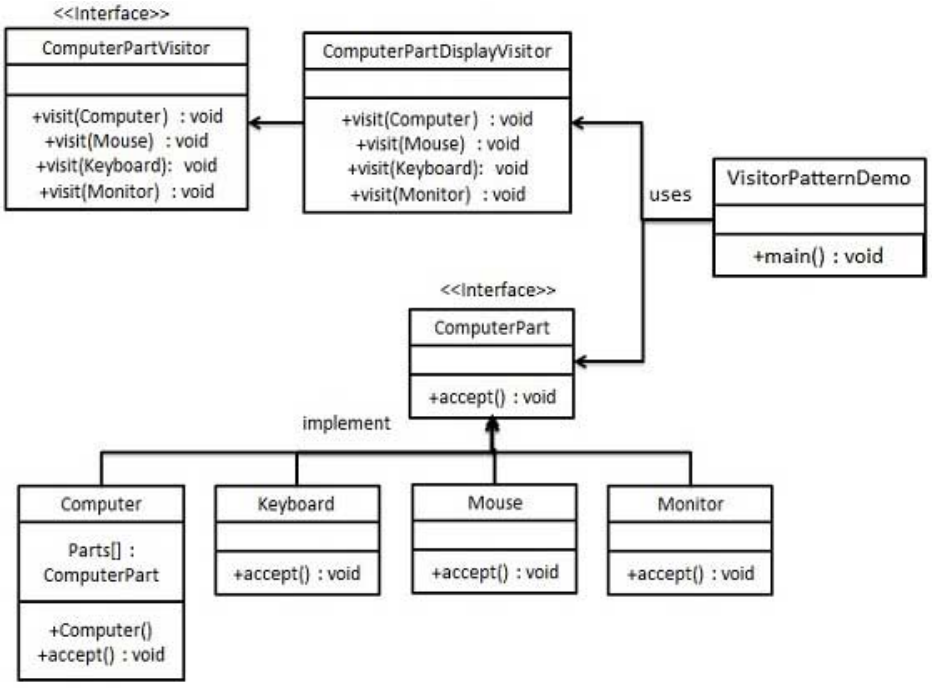

Figura I.20: Diagrama UML do Design Pattern Visitor