Engenharia de Software II

Aula: Diagrama de Classes.

Prof. Anderson Augusto Bosing

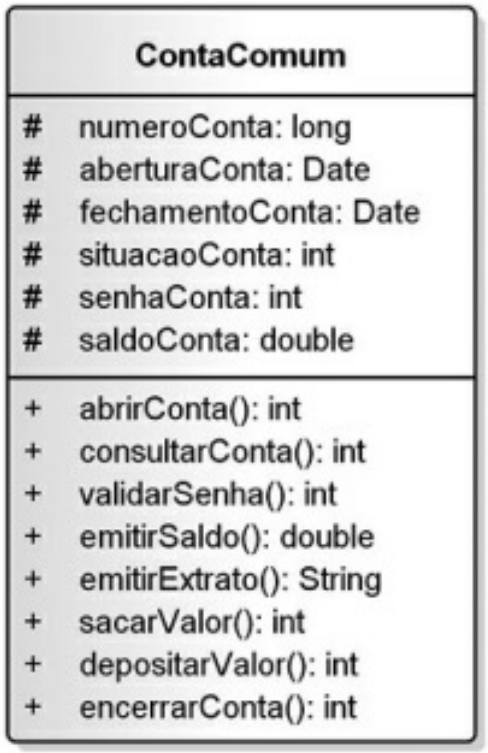
Diagrama de Classes

O diagrama de classes é um dos mais importantes e utilizados da UML.
Seu principal enfoque está em permitir a visualização das classes que comporão o sistema com seus respectivos atributos e métodos, bem como em demonstrar como as classes do diagrama se relacionam, complementam e transmitem informações entre si.
Esse diagrama apresenta uma visão estática de como as classes estão organizadas, preocupando-se em como definir a estrutura lógica delas.

Diagrama de Classes
Atributos e Métodos

Classes costumam ter atributos como já explicado anteriormente armazenam os dados dos objetos da classe.
Classes também costumam possuir métodos, também chamados operações, que são as funções que uma instância da classe pode executar.
Embora os métodos sejam declarados no diagrama de classes, identificando os possíveis parâmetros que são por eles recebidos e os possíveis valores por eles retornados, o diagrama de classes não se preocupa em definir as etapas que tais métodos deverão percorrer quando forem chamados.

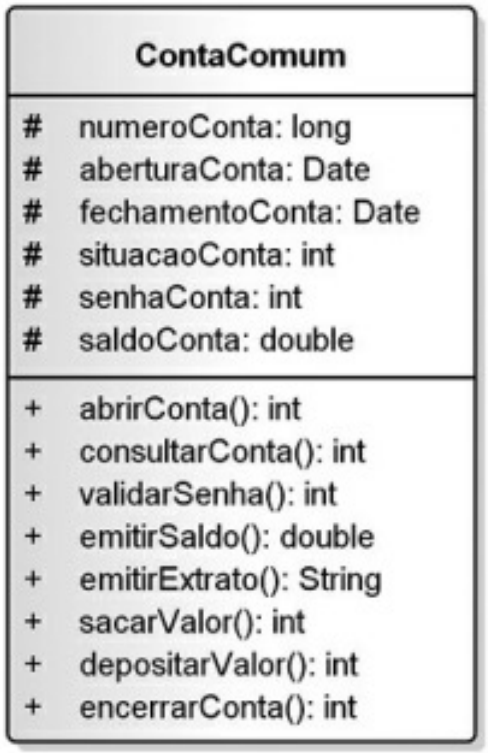
Diagrama de Classes
Classes



Uma classe, na linguagem UML, é representada como um retângulo com até três divisões.

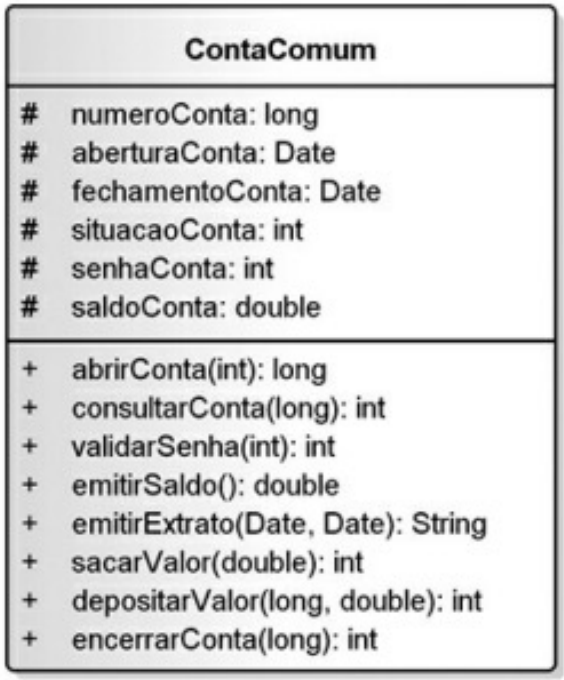
A primeira contém a descrição ou nome da classe, que, nesse exemplo, é ContaComum.

Diagrama de Classes
Classes



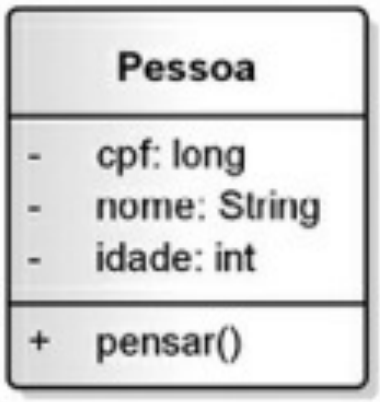
A segunda armazena os atributos e seus tipos de dados. A classe ContaComum contém os atributos numeroConta, do tipo long; aberturaConta e fechamentoConta, do tipo Date; situacaoConta e senhaConta, do tipo int; e saldoConta, do tipo double.

Diagrama de Classes
Classes



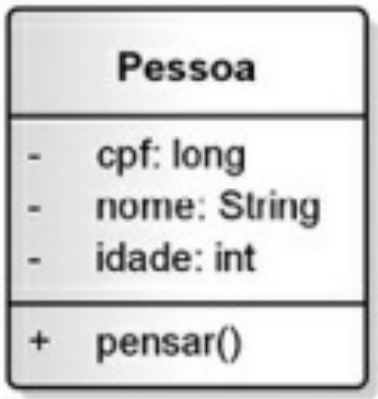
Finalmente, a terceira divisão lista os métodos da classe. Nesse exemplo, a classe ContaComum contém os métodos abrirConta, consultarConta, validarSenha, emitirSaldo, emitirExtrato, sacarValor, depositarValor e encerrarConta.
Os métodos são mapeados seguindo o padrão:
nomeMetodo(ParamEntrada):TipoRetorno

Diagrama de Classes
Tipos de Visibilidade



A visibilidade é utilizada para indicar o nível de acessibilidade de um determinado atributo ou método, sendo representada à esquerda destes.
Existem basicamente 3 modos de visibilidade: público, protegido, privado.

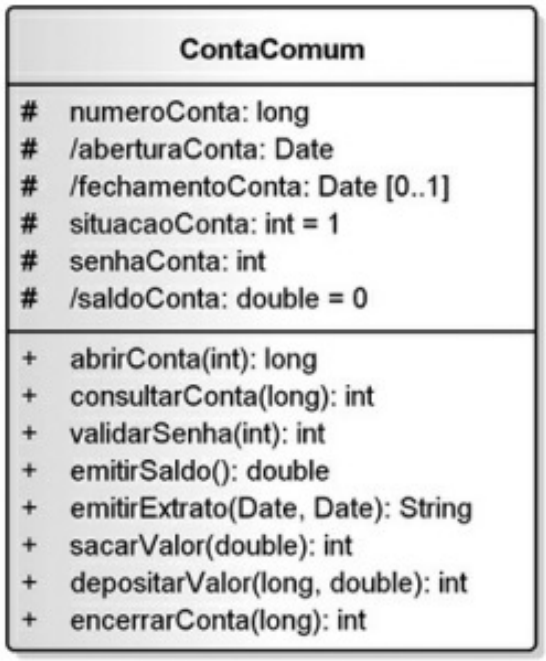
Diagrama de Classes
Tipos de Visibilidade



A visibilidade privada é representada por um símbolo de menos (-) e significa que somente os objetos da classe detentora do atributo ou método poderão enxergá-lo.

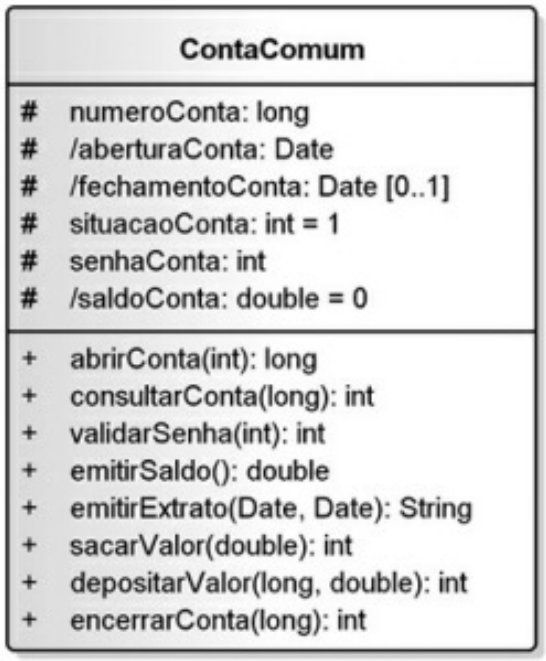
- A visibilidade protegida é representada pelo símbolo de sustenido (#) e determina que, além dos objetos da classe detentora do atributo ou método, também os objetos de suas subclasses poderão ter acesso a este.
- A visibilidade pública é representada por um símbolo de mais (+) e determina que o atributo ou método pode ser utilizado por qualquer objeto.

Diagrama de Classes
Características Extras dos Atributos



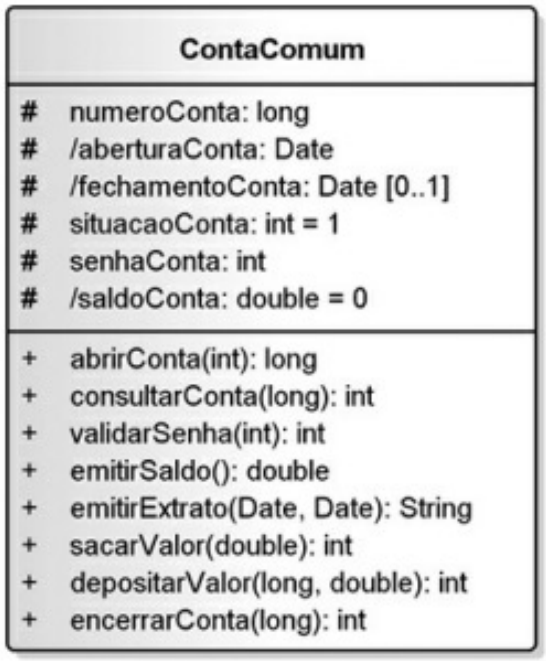
Os atributos aberturaConta, encerramentoConta e saldoConta têm uma barra (/) antes de seus nomes, significando que os valores desses atributos sofrem algum tipo de cálculo. No caso das datas, quando for realizada a operação de abertura de conta, o valor da data de abertura será tomado da data do sistema, o mesmo ocorrendo com o valor
da data de encerramento quando do encerramento da conta.

Diagrama de Classes
Características Extras dos Atributos



Pode-se perceber também que o atributo fechamentoConta, após a definição de seu tipo (Date), contém os valores [0..1]. Isso é chamado multiplicidade e, nesse contexto, significa que existirão, no mínimo,
nenhuma (0) e, no máximo, uma (1) data de encerramento

Diagrama de Classes
Características Extras dos Atributos



Também podemos verificar que o valor inicial dos atributos situacaoConta e saldoConta será, respectivamente, 1 e 0 quando da instanciação de um objeto dessa classe durante a abertura de uma nova conta. Dessa forma, sempre que uma nova conta for aberta, sua situação inicial terá o valor 1 (está ativa) e o seu saldo permanecerá com valor 0 até que um depósito seja realizado.

Vamos Praticar?

```java
public class Calculadora {

    private double resultado = 0;

    public double somar(double a, double b) {
        resultado = a + b;
        return resultado;
    }

    public double diminuir(double a, double b) {
        resultado = a - b;
        return resultado;
    }

    public double dividir(double a, double b) {
        resultado = a / b;
        return resultado;
    }

    public double multiplicar(double a, double b) {
        resultado = a * b;
        return resultado;
    }

}
```

Representar a Classe Calculadora no diagrama de classes.

https://online.visual-paradigm.com/

Vamos Praticar?

Representar a Classe Calculadora no diagrama de classes.

https://online.visual-paradigm.com/

Diagrama de Classes
Interfaces

Notações para representar interfaces na UML:
A primeira notação é a mesma para classes. São exibidas as operações que a interface especifica. Deve ser usado o estereótipo <<interface>>.
A segunda notação usa um segmento de reta com um pequeno círculo em um dos extremos e ligado ao classificador.
Classes clientes são conectadas à interface através de um relacionamento de notação similar à do relacionamento de dependência.

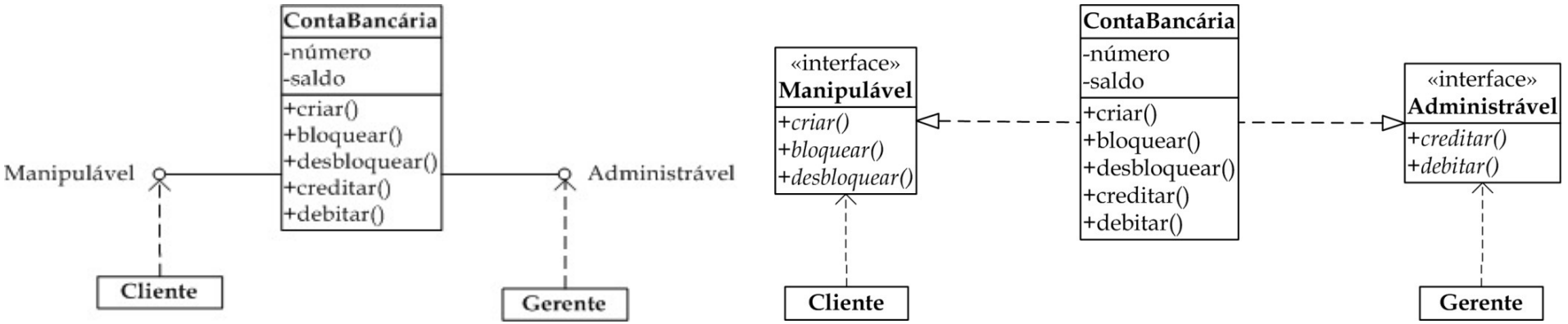
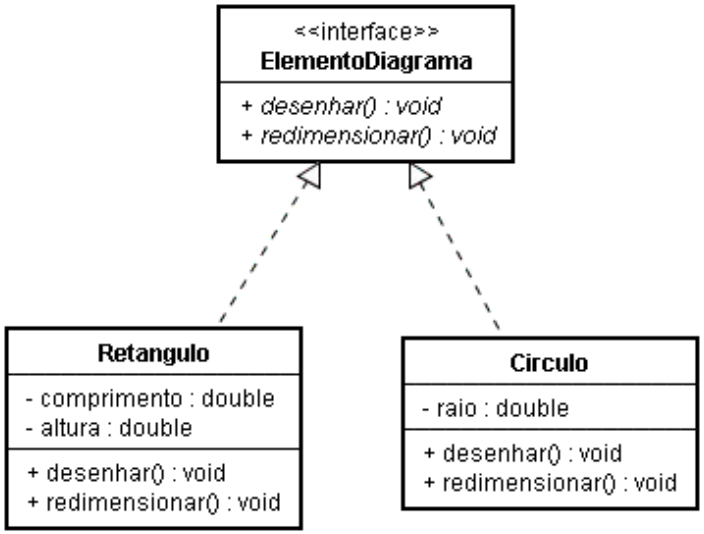


Diagrama de Classes Interfaces




```
public interface ElementoDiagrama {
    double PI = 3.1425926; //static and final
    constant.
    void desenhar();
    void redimensionar();
}

public class Circulo implements ElementoDiagrama
    {
    …
    public void desenhar() {  /* draw a circle*/ }
    public void redimensionar() { /* draw a circle*/ }
}

public class Retangulo implements
    ElementoDiagrama {
    …
    public void desenhar() { /* draw a rectangle*/  }
    public void redimensionar() { /* draw a
    rectangle*/ }
}
```

Chegou a vez de vocês?

Representar as Classes do projeto do repositório em um diagrama de classes.

https://github.com/profandersonbosing/unipar-diagram-class-1-main

https://online.visual-paradigm.com/

Perguntas?

Análise e Projetos de Sistemas

Aula: Diagrama de Classes(Relacionamentos ou Associações).

Prof. Anderson Augusto Bosing

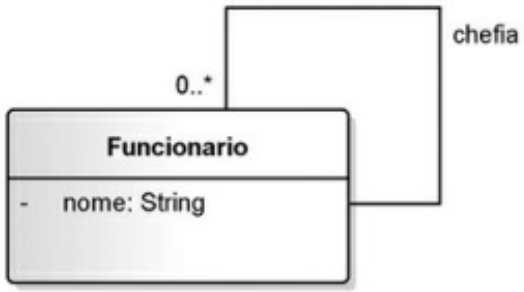
Relacionamentos ou Associações

As classes costumam ter relacionamentos entre si, chamados associações, que permitem que elas compartilhem informações entre si e colaborem para a execução dos processos executados pelo sistema. Uma associação descreve um vínculo que ocorre normalmente entre os objetos de uma ou mais classes. As associações são representadas por linhas ligando as classes envolvidas.

Tais linhas podem ter nomes ou títulos para auxiliar a compreensão do tipo de vínculo estabelecido entre os objetos das classes envolvidas nas associações.

Associação Unária ou Reflexiva

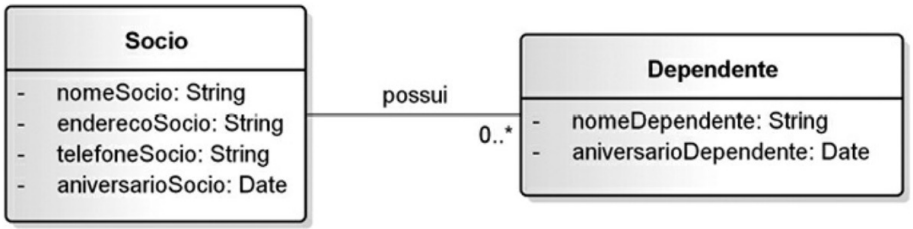
Este tipo de associação ocorre quando existe um relacionamento de um objeto de uma classe com objetos da mesma classe.



O título ou nome da associação não é realmente obrigatório, mas seu uso é recomendado para auxiliar a compreender o que a associação representa.
Normalmente é escolhido um verbo ou uma expressão verbal.

Associação Binária

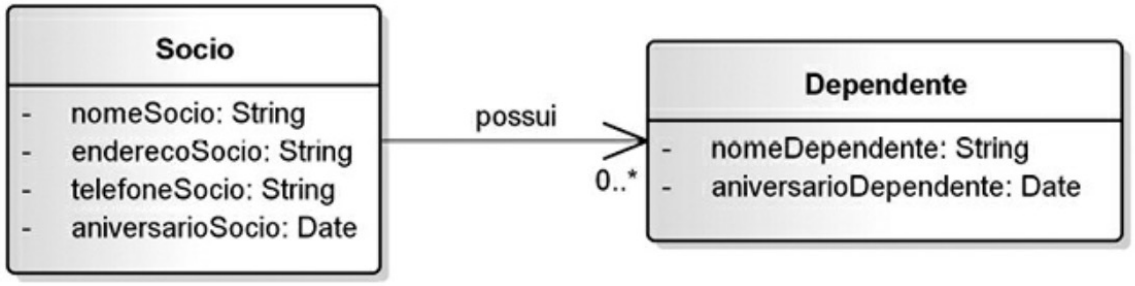
Associações binárias ocorrem quando são identificados relacionamentos entre objetos de duas classes distintas. Em geral, esse tipo de associação é o mais comumente encontrado.



Um objeto da classe Socio pode relacionar-se ou não com instâncias da classe Dependente

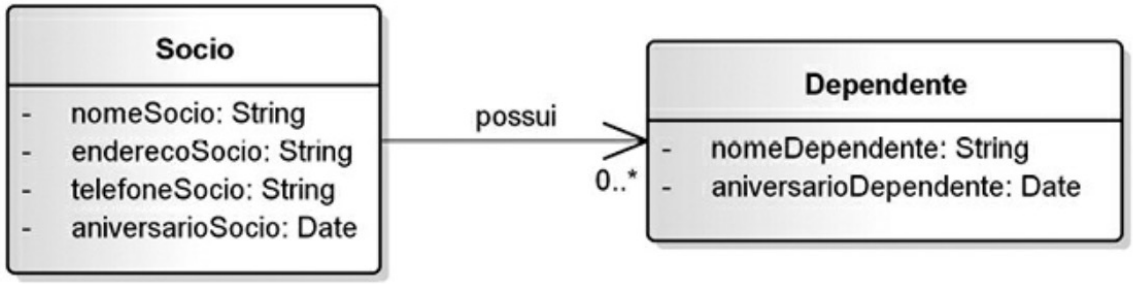
Associação Binária

A navegabilidade é representada mais comumente por uma seta em um dos fins da associação, embora seja possível representá-la nos dois sentidos, se isso for considerado necessário. Quando há navegabilidade unilateral, ela determina que os objetos da classe para onde a seta aponta não têm conhecimento dos objetos aos quais estão associados na outra extremidade da associação.



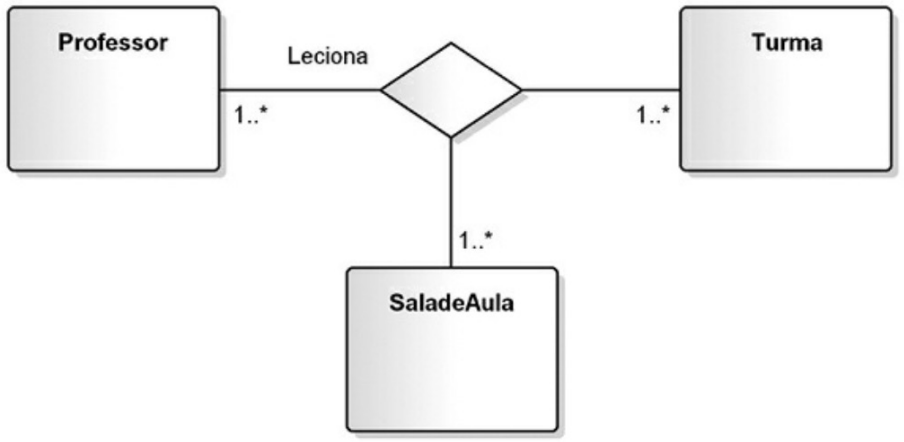
Associação Binária

No exemplo da figura abaixo, o uso da direção de leitura nos transmitiria a informação de que a leitura da associação deveria ser feita da seguinte forma: "Uma instância da classe Socio possui, no mínimo, nenhuma instância e, no máximo, muitas instâncias da classe Dependente e uma instância da classe Dependente é possuída por uma e somente uma instância da classe Socio".



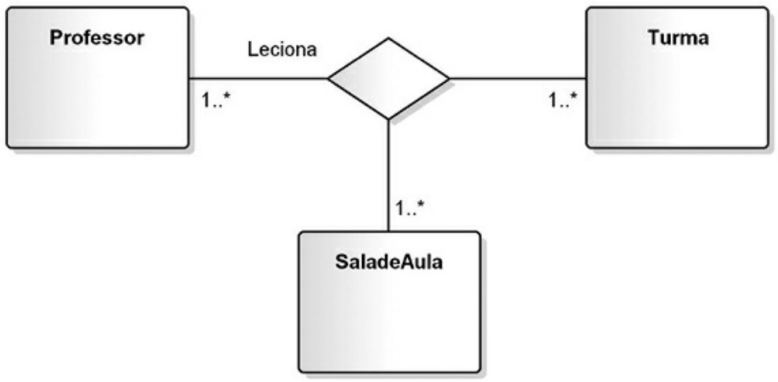
Associação Ternária ou N-ária

Associações ternárias ou n-árias conectam objetos de mais de duas classes. São representadas por um losango para onde convergem todas as ligações da associação. A figura abaixo apresenta um exemplo de associação ternária.



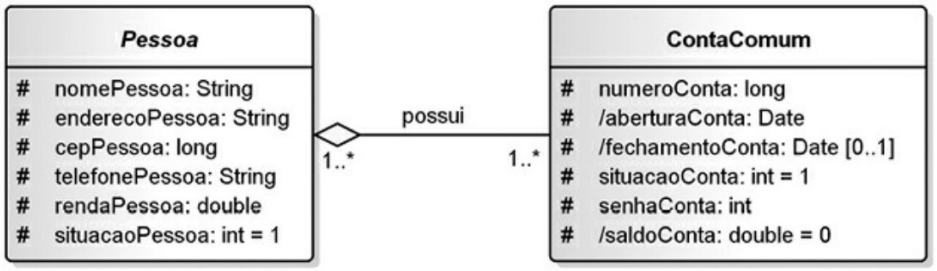
Associação Ternária ou N-ária

Assim, podemos ler a associação apresentada na figura da seguinte forma: “Um professor leciona para, no mínimo, uma turma e, no máximo, para muitas, uma turma tem, no mínimo, um professor e, no máximo, muitos lecionando para ela e um professor, ao lecionar para uma determinada turma, utiliza, no mínimo, uma sala de aula e, no máximo, muitas”. As associações ternárias são úteis para demonstrar associações complexas. No entanto, convém evitar utilizá-las, pois sua leitura é, por vezes, difícil de ser interpretada, todavia seu uso pode ser inevitável em algumas situações.



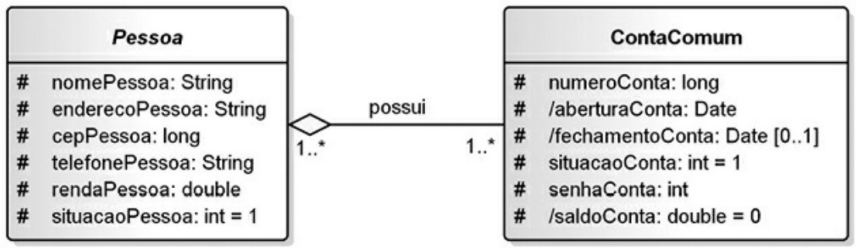
Agregação

Agregação é um tipo especial de associação em que se tenta demonstrar que as informações de um objeto (objeto-todo) são complementadas pelas informações contidas em um ou mais objetos no outro fim da associação (chamados objetos-parte). Esse tipo de associação tenta demonstrar uma relação todo/parte entre os objetos associados. O símbolo de agregação difere do de associação por conter um losango no fim da associação que contém os objetos-todo.



Agregação

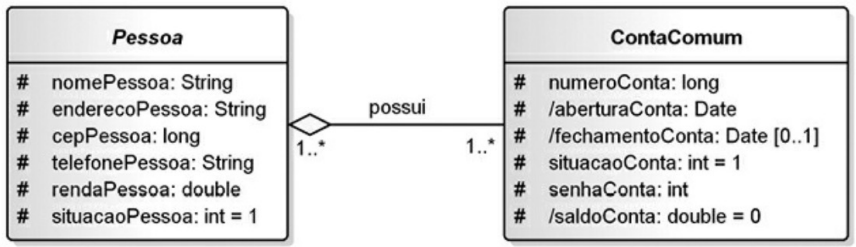
Esse exemplo demonstra uma associação de agregação existente entre uma classe Pessoa e uma classe ContaComum, o que determina que os objetos da classe Pessoa são objetos-todo que precisam ter suas informações complementadas pelos objetos da classe ContaComum, que, nessa associação, são objetos-parte. Dessa maneira, sempre que uma pessoa for consultada, além das informações pessoais, serão apresentadas todas as contas que ela possui.



A decisão de utilizar a AGREGAÇÃO é uma questão de julgamento. Nem sempre é evidente que uma associação deve ser modelada como uma AGREGAÇÃO.

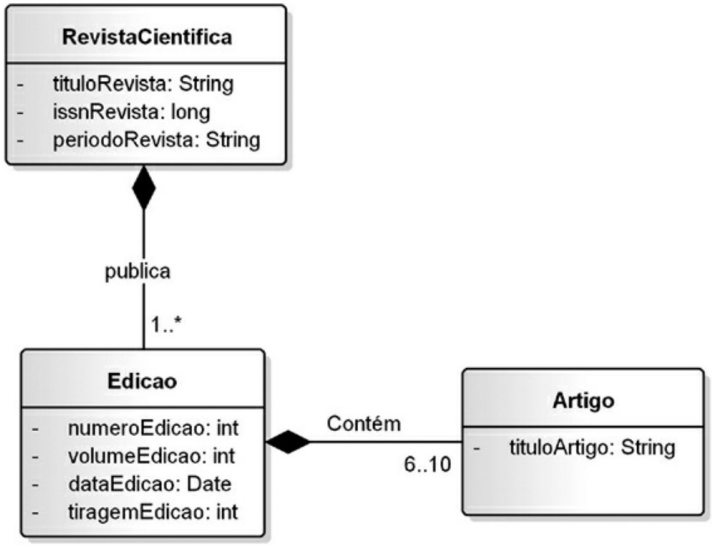
Agregação

A associação de agregação pode, em alguns casos, ser substituída por uma associação binária simples, dependendo da visão de quem faz a modelagem. A função principal de uma associação do tipo agregação é identificar a obrigatoriedade de uma complementação das informações de um objeto-todo por seus objetos-parte, quando este for consultado. Em uma associação binária, todavia, essa obrigatoriedade não está explícita.



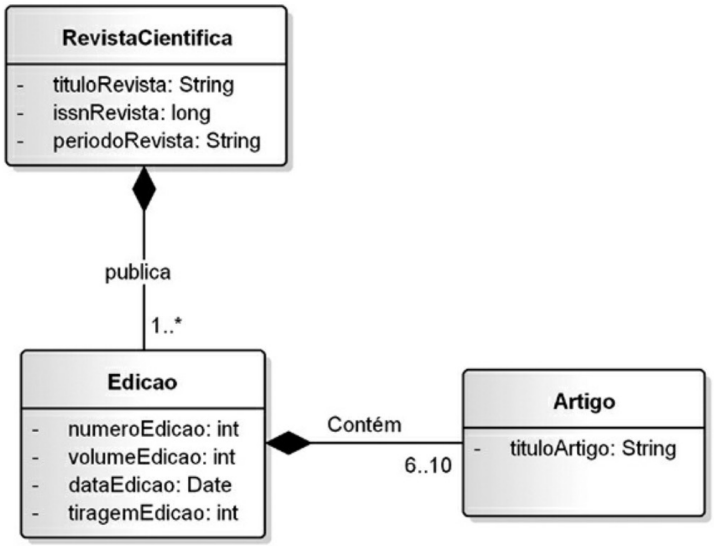
Composição

Uma associação do tipo composição constitui-se em uma variação da agregação, onde é apresentado um vínculo mais forte entre os objetos-todo e os objetos-parte, procurando demonstrar que os objetos-parte têm de estar associados a um único objeto-todo. Em uma composição, os objetos- parte não podem ser destruídos por um objeto diferente do objeto-todo ao qual estão relacionados.



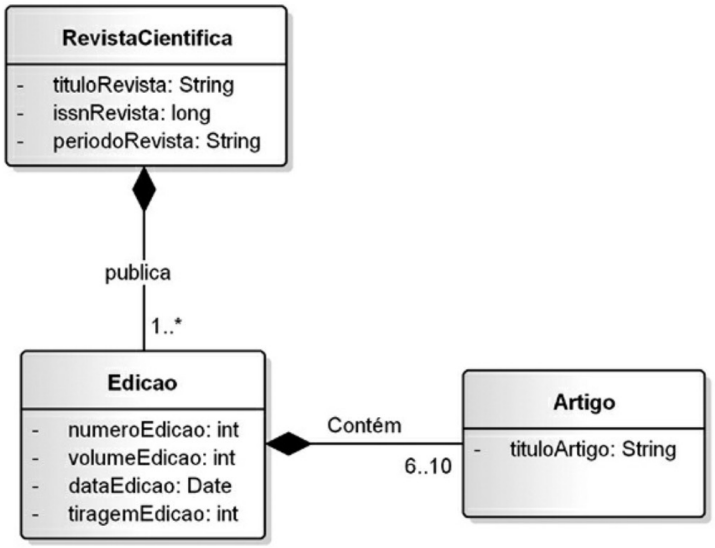
Composição

É possível perceber que um objeto da classe RevistaCientifica refere-se a, no mínimo, um objeto da classe Edicao, podendo se referir a muitos objetos dessa classe, e cada instância da classe Edicao relaciona-se única e exclusivamente a uma instância específica da classe RevistaCientifica, não podendo relacionar-se a nenhuma outra.



Composição

Verificando a imagem, é possível perceber que um objeto da classe RevistaCientifica refere-se a, no mínimo, um objeto da classe Edicao, podendo se referir a muitos objetos dessa classe, e cada instância da classe Edicao relaciona-se única e exclusivamente a uma instância específica da classe RevistaCientifica, não podendo relacionar-se a nenhuma outra.
Ainda nesse exemplo, percebemos que um objeto da classe Edicao deve se relacionar a, no mínimo, seis objetos da classe Artigo, podendo se relacionar com até 10 objetos da já citada classe. Esse tipo de informação torna-se útil como documentação e serve como forma de validação, que impede que uma revista seja publicada sem ter, no mínimo, seis artigos ou mais de 10. No entanto, um objeto da classe Artigo refere-se unicamente a um objeto da classe Edicao. Isso é também uma forma de documentação, pois uma edição de uma revista científica só deve publicar trabalhos inéditos. Assim, é lógico que não é possível a um mesmo objeto da classe Artigo relacionar-se a mais de um objeto da classe Edicao.



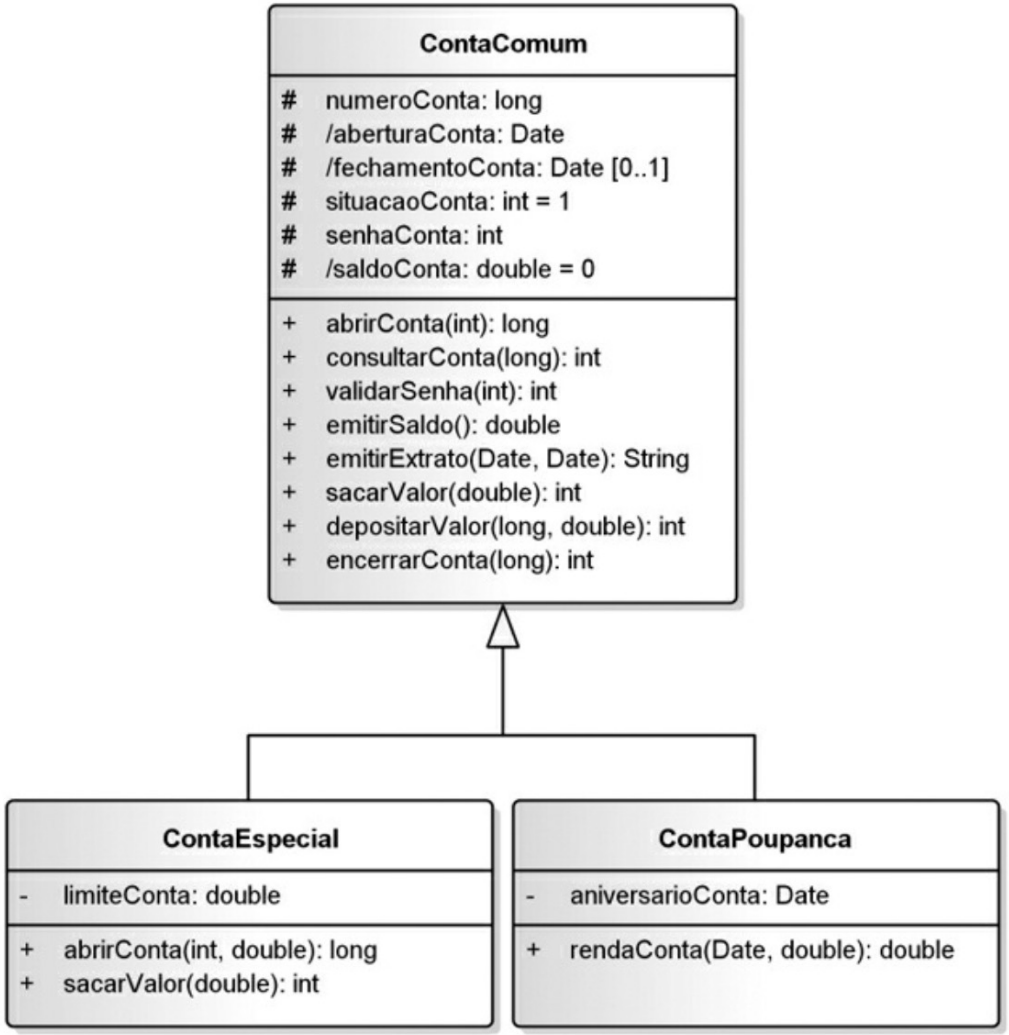
Generalização/Especialização

Este é um tipo especial de relacionamento, similar à associação de mesmo nome utilizada no diagrama de casos de uso. O objetivo dessa associação é representar a ocorrência de herança entre as classes, identificando as classes-mãe (ou superclasses), chamadas gerais, e classes-filhas (ou subclasses), chamadas especializadas, demonstrando a hierarquia entre as classes e, possivelmente, métodos polimórficos nas classes especializadas.

Generalização/Especialização

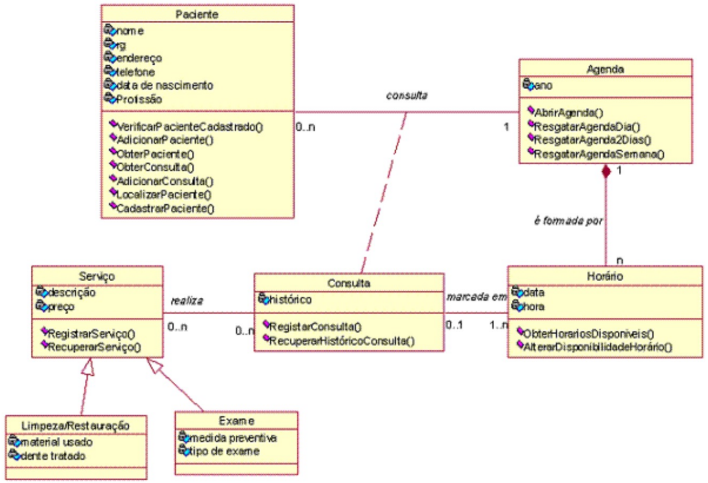
Esse tipo de relacionamento permite representar classes derivadas a partir de classes mais antigas e, ao mesmo tempo que as novas classes herdam todos os atributos e métodos das classes das quais foram derivadas, é possível adicionar novos atributos e/ou métodos a essas classes, dessa forma especializando-as. Isso permite maior rapidez no desenvolvimento, uma vez que não é necessário adicionar os atributos e métodos já existentes às classes anteriores, apenas os novos atributos ou métodos, o que também impede que erros de codificação sejam cometidos desnecessariamente. Além disso, métodos podem ser redeclarados em uma classe especializada, com o mesmo nome, mas comportando-se de forma diferente, não sendo, portanto, necessário modificar o código-fonte do sistema em relação às chamadas de métodos das classes especializadas, pois o nome do método não mudou, somente foi redeclarado em uma classe especializada e só se comportará de maneira diferente quando for chamado por objetos dessa classe.

Generalização/Especialização



Chegou a vez do vocês!

https://docs.google.com/document/dz1dIF__XFcNEehfwREJnX1BWgP-4VDgBuhstQyUbiGBdw/edit?usp=sharing



Técnicas para identificação de classes

Várias técnicas (de uso não exclusivo) são usadas para identificar classes:

Categorias de Conceitos

Análise Textual de Abbott (Abbot Textual Analysis)

Categorização BCE

Padrões de Análise (Analisys Patterns)

Identificação Dirigida a Responsabilidades

Categorias de Conceitos

Estratégia: usar uma lista de conceitos comuns.
Conceitos concretos. Por exemplo, edifícios, carros, salas de aula, etc.
Papéis desempenhados por seres humanos. Por exemplo, professores, alunos, empregados, clientes, etc.
Eventos, ou seja, ocorrências em uma data e em uma hora particulares. Por exemplo, reuniões, pedidos, aterrisagens, aulas, etc.
Lugares: áreas reservadas para pessoas ou coisas. Por exemplo: escritórios, filiais, locais de pouso, salas de aula, etc.
Organizações: coleções de pessoas ou de recursos. Por exemplo: departamentos, projetos, campanhas, turmas, etc.
Conceitos abstratos: princípios ou idéias não tangíveis. Por exemplo: reservas, vendas, inscrições, etc.

Análise Textual de Abbott

Estratégia: identificar termos da narrativa de casos de uso e documento de requisitos que podem sugerir classes, atributos, operações.

Neste técnica, são utilizadas diversas fontes de informação sobre o sistema: documento e requisitos, modelos do negócio, glossários, conhecimento sobre o domínio, etc.

Para cada um desses documentos, os nomes (substantivos e adjetivos) que aparecem no mesmo são destacados. (São também consideradas locuções equivalentes a substantivos.)

Após isso, os sinônimos são removidos (permanecem os nomes mais significativos para o domínio do negócio em questão).

Análise Textual de Abbott

Cada termo remanescente se encaixa em uma das situações a seguir:
O termo se torna uma classe (ou seja, são classes candidatas);
O termo se torna um atributo;
O termo não tem relevância alguma com ao SSOO.
Abbott também preconiza o uso de sua técnica na identificação de operações e de associações.
Para isso, ele sugere que destaquemos os verbos no texto.
Verbos de ação (e.g., calcular, confirmar, cancelar, comprar, fechar, estimar, depositar, sacar, etc.) são operações em potencial.
Verbos com sentido de "ter" são potenciais agregações ou composições.
Verbos com sentido de "ser" são generalizações em potencial.
Demais verbos são associações em potencial.

Análise Textual de Abbott

Cada termo remanescente se encaixa em uma das situações a seguir:
O termo se torna uma classe (ou seja, são classes candidatas);
O termo se torna um atributo;
O termo não tem relevância alguma com ao SSOO.
Abbott também preconiza o uso de sua técnica na identificação de operações e de associações.
Para isso, ele sugere que destaquemos os verbos no texto.
Verbos de ação (e.g., calcular, confirmar, cancelar, comprar, fechar, estimar, depositar, sacar, etc.) são operações em potencial.
Verbos com sentido de "ter" são potenciais agregações ou composições.
Verbos com sentido de "ser" são generalizações em potencial.
Demais verbos são associações em potencial.

Análise Textual de Abbott

Parte do Texto	Componente	Exemplo
Nome próprio	Objeto	Eduardo Bezerra
Nome simples	Classe	aluno
Verbos de Ação	Operação	registrar
Verbo Ser	Herança	é um
Verbo Ter	Todo-parte	tem um

Análise Textual de Abbott

A ATA é de aplicação bastante simples.
No entanto, uma desvantagem é que seu resultado (as classes candidatas identificadas) depende de os documentos utilizados como fonte serem completos.
Dependendo do estilo que foi utilizado para escrever esse documento, essa técnica pode levar à identificação de diversas classes candidatas que não gerarão classes.
A análise do texto de um documento pode não deixar explícita uma classe importante para o sistema.
Em linguagem natural, as variações lingüísticas e as formas de expressar uma mesma idéia são bastante numerosas

Análise de Casos de Uso

Essa técnica é também chamada de identificação dirigida por casos de uso, e é um caso particular da ATA.
Técnica preconizada pelo Processo Unificado.
Nesta técnica, o MCU é utilizado como ponto de partida.
Premissa: um caso de uso corresponde a um comportamento específico do SSOO. Esse comportamento somente pode ser produzido por objetos que compõem o sistema.
Em outras palavras, a realização de um caso de uso é responsabilidade de um conjunto de objetos que devem colaborar para produzir o resultado daquele caso de uso.
Com base nisso, o modelador aplica a técnica de análise dos casos de uso para identificar as classes necessárias à produção do comportamento que está documentado na descrição do caso de uso.

Análise de Casos de Uso

Procedimento de aplicação:
O modelador estuda a descrição textual de cada caso de uso para identificar classes candidatas.
Para cada caso de uso, se texto (fluxos principal, alternativos e de exceção, pós-condições e pré-condições, etc.) é analisado.
Na análise de certo caso de uso, o modelador tenta identificar classes que possam fornecer o comportamento do mesmo.
Na medida em que os casos de uso são analisados um a um, as classes do SSOO são identificadas.
Quando todos os casos de uso tiverem sido analisados, todas as classes (ou pelo menos a grande maioria delas) terão sido identificadas.
Na aplicação deste procedimento, podemos utilizar as categorização BCE.

Categorização BCE

Na categorização BCE, os objetos de um SSOO são agrupados de acordo com o tipo de responsabilidade a eles atribuída.

- objetos de entidade: usualmente objetos do domínio do problema
- objetos de fronteira: atores interagem com esses objetos
- objetos de controle: servem como intermediários entre objetos de fronteira e de entidade, definindo o comportamento de um caso de uso específico.

Categorização proposta por Ivar Jacobson em1992.

Possui correspondência (mas não equivalência!) com o framework model-view-controller (MVC).

Ligação entre análise (o que; problema) e projeto (como; solução)

Estereótipos na UML: «boundary», «entity», «control»

Objetos de Entidade

Repositório para informações e as regras de negócio manipuladas pelo sistema.
Representam conceitos do domínio do negócio.
Características
Normalmente armazenam informações persistentes.
Várias instâncias da mesma entidade existindo no sistema.
Participam de vários casos de uso e têm ciclo de vida longo.
Exemplo:
Um objeto Pedido participa dos casos de uso Realizar Pedido e Atualizar Estoque. Este objeto pode existir por diversos anos ou mesmo tanto quanto o próprio sistema.

Objetos de Fronteira

Realizam a comunicação do sistema com os atores.
traduzem os eventos gerados por um ator em eventos relevantes ao sistema eventos de sistema.
também são responsáveis por apresentar os resultados de uma interação dos objetos em algo inteligível pelo ator.
Existem para que o sistema se comunique com o mundo exterior.
Por conseqüência, são altamente dependentes do ambiente.
Há dois tipos principais de objetos de fronteira:
Os que se comunicam com o usuário (atores humanos): relatórios, páginas HTML, interfaces gráfica desktop, etc.
Os que se comunicam com atores não-humanos (outros sistemas ou dispositivos): protocolos de comunicação.

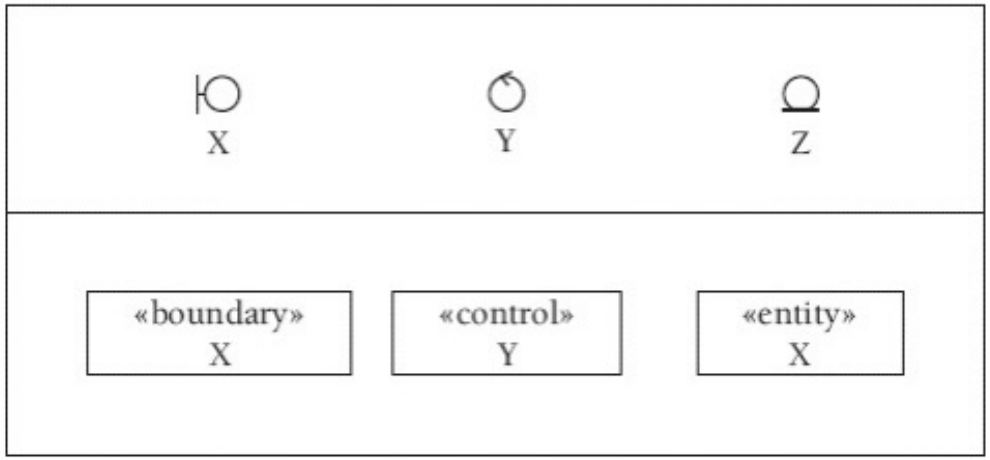
Objetos de Controle

São a “ponte de comunicação” entre objetos de fronteira e objetos de entidade.
Responsáveis por controlar a lógica de execução correspondente a um caso de uso.
Decidem o que o sistema deve fazer quando um evento de sistema ocorre.
Eles realizam o controle do processamento
Agem como gerentes (coordenadores, controladores) dos outros objetos para a realização de um caso de uso.
Traduzem eventos de sistema em operações que devem ser realizadas pelos demais objetos.

Importância da Categorização BCE

A categorização BCE parte do princípio de que cada objeto em um SSOO é especialista em realizar um de três tipos de tarefa, a saber:
se comunicar com atores (fronteira),
manter as informações (entidade) ou
coordenar a realização de um caso de uso (controle).
A categorização BCE é uma "receita de bolo" para identificar objetos participantes da realização de um caso de uso.
A importância dessa categorização está relacionada à capacidade de adaptação a eventuais mudanças.
Se cada objeto tem atribuições específicas dentro do sistema, mudanças podem ser menos complexas e mais localizadas.
Uma modificação em uma parte do sistema tem menos possibilidades de resultar em mudanças em outras partes.

Notação da UML para objetos, segunda a categorização BCE



Atividade

Além do diagrama de classes, a UML define outro tipo de diagrama estrutural: o diagrama de objetos.

O que é ?
Qual seu principal objetivo ?
Quais são seus principais componentes ?
Quais suas vantagens ?

Arquitetura Física

A arquitetura física (ou arquitetura de implantação) diz respeito à disposição dos subsistemas de um SSOO pelos nós de processamentos disponíveis. Para sistemas simples, que executam em um único nó de processamento, a definição da arquitetura de implantação não faz sentido. No entanto, para sistemas mais complexos, é fundamental conhecer quais são os componentes físicos do sistema, quais são as interdependências entre eles e de que forma as camadas lógicas do sistema são dispostas por esses componentes.

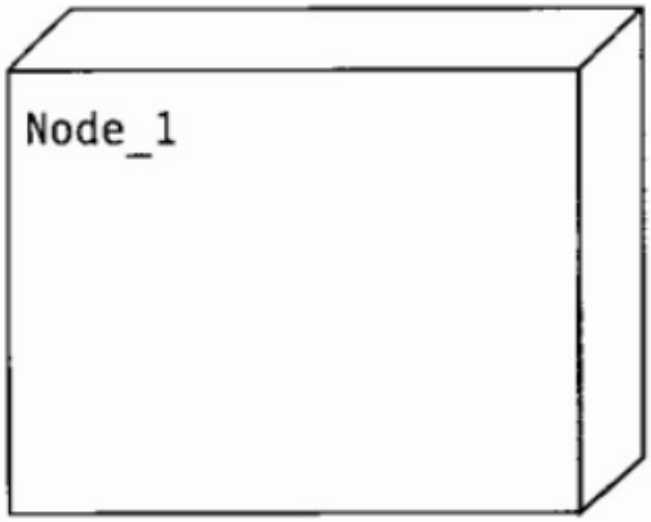
Diagrama de Implantação

O diagrama de implantação representa como é realizada a distribuição do sistema através de nós de hardware, componentes e dependências de software e suas devidas relações de comunicação.

Um diagrama de implantação modela o inter-relacionamento entre os recursos de infraestrutura de rede ou artefatos de sistema. Normalmente representamos servidores neste diagrama. Estes recursos são chamados de nodes ou nós.

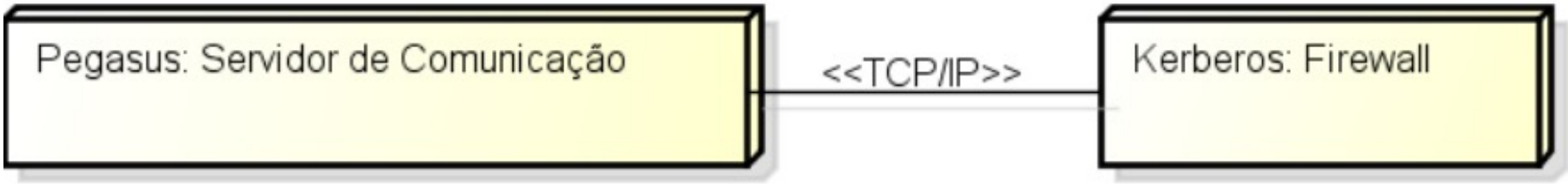
Diagrama de Implantação

Cada nó é uma máquina física que encerra um ou vários componentes. Outros dispositivos podem ser representados com o estereótipo de <<dispositivos>> ou <<device>>



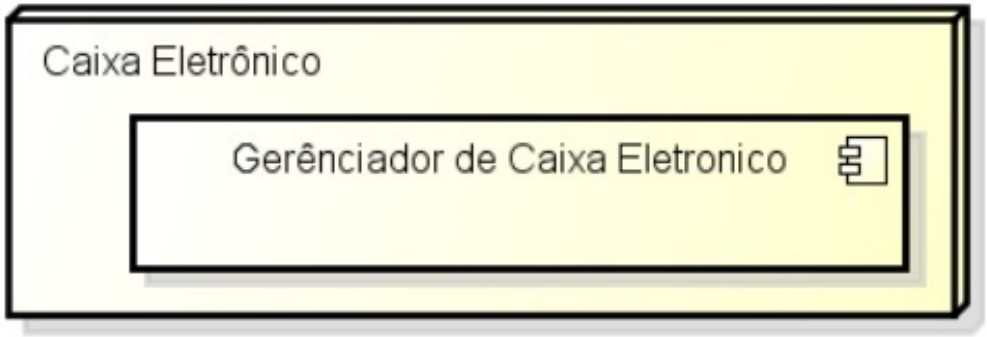
Associações

Os nós podem possuir ligações entre si de forma que possam se comunicar e trocar informações.

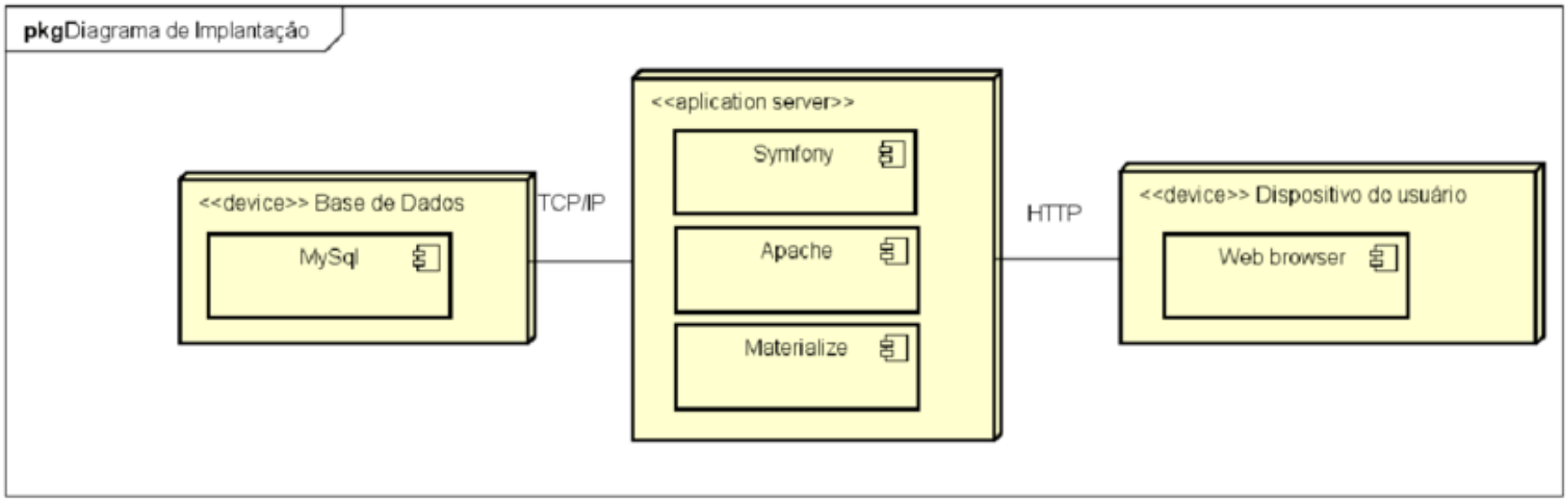


Nós em Componentes

Comum identificar os componentes que são executados por um nós.



Exemplo


pkgDiagrama de Implantação
<<aplication server>>
Symfony
Apache
Materialize
<<device>> Base de Dados
MySql
TCP/IP
HTTP
<<device>> Dispositivo do usuário
Web browser

Arquitetura do Sistema

Um sistema orientado a objetos(SSOO) é composto de objetos que interagem entre si por meio de envio de mensagens com objetivo de executar tarefas.

Cada um desses objetos se comporta de acordo com a definição de sua classe. Por outra perspectiva, um SSOO também pode ser visto como um conjunto de subsistemas que o compõem.

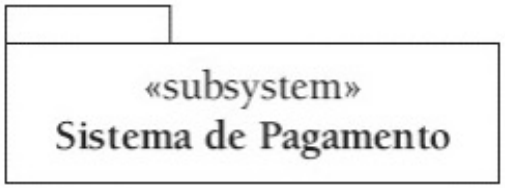
A definição dos subsistemas de um SSOO é feita no projeto da arquitetura, ou projeto arquitetural.

Arquitetura do Sistema

Atualmente, não há uma definição universal quanto ao que significa arquitetura de software. De acordo com o documento de especificação da UML (OMG, 2001), a definição desse termo é a seguinte: *É a estrutura organizacional do software. Uma arquitetura pode ser recursivamente decomposta em partes que interagem através de interfaces. Relacionamentos conectam as partes e restrições que se aplicam ao agrupamento das partes*. O termo *recursivamente* nessa definição indica que um sistema é composto de partes, sendo que as próprias partes são também sistemas relativamente independentes que cooperam entre si para realizar as tarefas do sistema.

Arquitetura Lógica

Chamamos de arquitetura lógica à organização das classes de um SSOO em subsistemas. Mas o que é um subsistema?
Um sistema de software orientado a objetos, como todo sistema, pode ser subdividido em subsistemas, cada um dos quais corresponde a um aglomerado de classes e de interfaces. Um subsistema provê serviços para outros por meio de sua interface, que corresponde a um conjunto de serviços que ele provê. Cada subsistema provê ou utiliza serviços de outros subsistemas.



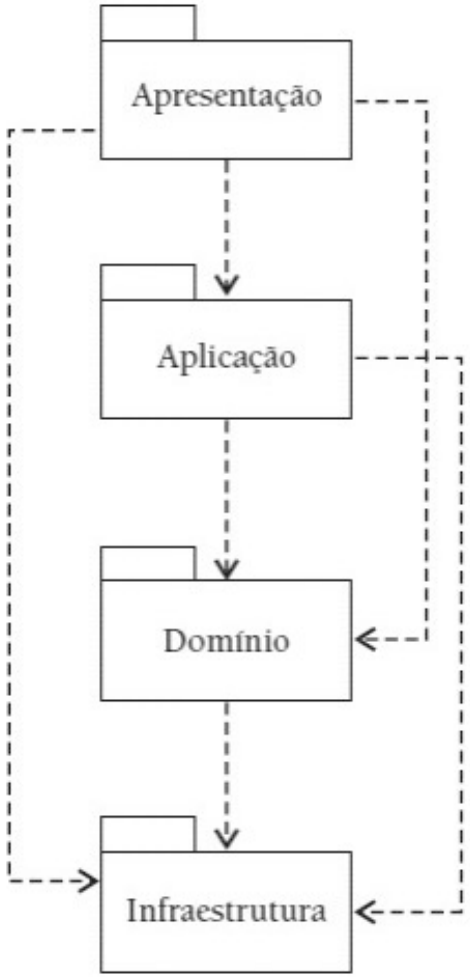
Conceito de Camada de Software

Uma camada de software (ou simplesmente camada) é uma coleção de unidades de software (como classes ou componentes) que podem ser executadas ou acessadas. As camadas representam diferentes níveis de abstração. Dessa forma, um SSOO é representado por uma pilha de camadas de software que se comunicam entre si. As camadas inferiores representam serviços cada vez mais genéricos (que podem ser utilizados em diversos sistemas), enquanto as superiores representam serviços cada vez mais específicos ao sistema em questão.

Camadas Típicas de um SI

A nomenclatura utilizada pela literatura para nomear as camadas típicas de um sistema de informação está longe de uma padronização. Entretanto, uma divisão tipicamente encontrada para as camadas lógicas de um SSOO é a que separa o sistema nas seguintes camadas: apresentação, aplicação, domínio e infraestrutura. Nessa lista de nomes, da esquerda para a direita, temos camadas cada vez mais genéricas. Também da esquerda para a direita temos a ordem de dependência entre as camadas; por exemplo, a camada da apresentação depende (requisita serviços) da camada de aplicação, mas não o contrário.

Camadas Típicas de um SI



Camada da apresentação

Essa camada também é conhecida como camada de interface com o usuário. A camada da apresentação é composta de classes que constituem a funcionalidade para visualização dos dados pelos usuários e interface com outros sistemas. As classes de fronteira se encontram nessa camada.
Exemplos de camadas de apresentação:
um sistema de menus baseados em texto;
uma interface gráfica construída em algum ambiente de programação.

Os objetos de fronteira que interagem com usuários são alocados nesta camada.

Camada da apresentação

A camada da apresentação deve apenas servir como um ponto de captação de informações a partir do ambiente, ou de apresentação de informações que o sistema processou. Portanto, não devemos atribuir às classes dessa camada responsabilidades relativas à lógica do negócio. A única inteligência que essas classes devem ter é a que lhes permite realizar a comunicação com o ambiente do sistema. Esse tipo de lógica por vezes é denominado lógica da apresentação.

As principais formas de interação com o ambiente que podem existir em um sistema de software

Clientes WEB.
Clientes (dispositivos) móveis.
Clientes stand-alone.
Serviços WEB.

Camada da aplicação

Essa camada também é conhecida como camada de serviço. A camada da aplicação é a que serve de intermediária entre os vários componentes da camada de apresentação (p. ex., telas da interface gráfica e interfaces de voz, interfaces por linha de comando etc) e a lógica contida nos objetos do negócio. Esta camada traduz as mensagens que recebe da camada da aplicação em mensagens compreendidas pelos objetos do domínio. É também responsabilidade dessa camada o controle da navegação do usuário de uma janela a outra, quando a interação ocorrer por meio de uma interface gráfica.

Os objetos de controle são alocados a essa camada

Camada da aplicação

Essa camada também é conhecida como camada de serviço. A camada da aplicação é a que serve de intermediária entre os vários componentes da camada de apresentação (p. ex., telas da interface gráfica e interfaces de voz, interfaces por linha de comando etc) e a lógica contida nos objetos do negócio. Esta camada traduz as mensagens que recebe da camada da aplicação em mensagens compreendidas pelos objetos do domínio. É também responsabilidade dessa camada o controle da navegação do usuário de uma janela a outra, quando a interação ocorrer por meio de uma interface gráfica.

Os objetos de controle são alocados a essa camada

Camada do domínio

Também é chamada de camada do negócio. Esta camada recebe requisições provenientes da camada da aplicação. Os objetos nesta camada normalmente são independentes da aplicação (o que significa que podem ser reusados em diferentes aplicações dentro de uma corporação). Essa camada é responsável por validações das regras de negócio assim como de dados provenientes da camada de apresentação (por intermédio da camada de aplicação).

As classes residentes na camada do domínio são aquelas que representam os conceitos do domínio da aplicação que o sistema deve processar. Essas classes representam as informações produzidas durante a realização dos processos do negócio, assim como as regras de negócio que direcionam a manipulação dessas informações. Elas também possuem uma denominação alternativa, classes do negócio

Camada de infraestrutura

Também é chamada de camada do negócio. Esta camada recebe requisições provenientes da camada da aplicação. Os objetos nesta camada normalmente são independentes da aplicação (o que significa que podem ser reusados em diferentes aplicações dentro de uma corporação). Essa camada é responsável por validações das regras de negócio assim como de dados provenientes da camada de apresentação (por intermédio da camada de aplicação).

As classes residentes na camada do domínio são aquelas que representam os conceitos do domínio da aplicação que o sistema deve processar. Essas classes representam as informações produzidas durante a realização dos processos do negócio, assim como as regras de negócio que direcionam a manipulação dessas informações. Elas também possuem uma denominação alternativa, classes do negócio

Camada de infraestrutura

Essa camada é também conhecida como camada de serviços técnicos. A camada de infraestrutura é o lugar onde são encontrados serviços genéricos e úteis para uma gama bastante vasta de aplicações. Como o próprio nome diz, essa camada fornece serviços relacionados às tecnologias usadas implementação da aplicação. Exemplos de serviços fornecidos por esta camada são autenticação e autorização, controle de transações, registro de operações do sistema (logging), manipulação de arquivos, estruturas de dados (vetores de caracteres, mapas, listas etc.), classes utilitárias, entre outros. Além dos serviços listados acima, a camada de infraestrutura também deve conter classes cujo objetivo é permitir que o sistema se comunique com outros sistemas para realizar tarefas ou adquirir informações (p. ex., acesso a um SGBD ou a serviços WEB)

O padrão MVC e sua relação com a arquitetura lógica

Anteriormente visualizamos as camadas tipicamente encontradas em sistemas de informação. Em particular, mencionamos que, idealmente, a camada da apresentação não deve conter inteligência, mas sim implementar apenas lógica da apresentação (preenchimento de controles com dados provenientes da camada da aplicação, habilitação de controles, definição de cores etc.). Um padrão arquitetural normalmente utilizado para alcançar essa separação de responsabilidades entre a lógica da apresentação e a lógica da aplicação é o Model-View-Controller (MVC).

O padrão MVC e sua relação com a arquitetura lógica

O Model-View-Controller é um padrão de software que descreve a interação entre objetos da interface com o usuário e os demais objetos de uma aplicação. Em particular, esse padrão propõe:
(1) uma forma de organizar a interface com o usuário;
(2) descreve de que maneira atualizar o estado dessa interface.
A proposta inicial desse padrão foi apresentada na linguagem de programação Smalltalk-80. Ao longo do tempo, diversas variações da proposta original foram proposta. De forma geral, todas as variantes apresentam três componentes, Model, View e Controller

Model

Esse componente é a parte da aplicação que contém os dados e suas validações. Em outras palavras, o componente Model corresponde ao estado, à estrutura e ao comportamento dos dados sendo visualizados e manipulados pelo usuário da aplicação por meio da interface gráfica. Esse componente disponibiliza operações em sua interface para que o restante da aplicação possa manipulá-lo (p. ex., consultá-lo ou editá-lo).

View

Uma aplicação pode apresentar aos seus usuários diversas perspectivas (visões) de uma mesma informação (ou de um mesmo modelo). Por exemplo, pode haver uma interface gráfica para editar as notas de um aluno em uma turma, assim como outra interface gráfica para permitir a visualização dessas notas para impressão apenas. O componente View do MVC representa cada uma das possíveis formas de apresentar uma informação proveniente do componente Model.

Controller

Cada visão possui a ela associada um objeto controlador, que auxilia na implementação da interface gráfica associada à visão correspondente.

Trabalho(Valor = 3,0 Pontos)

Realizar trabalho de pesquisa sobre:
1 - Modelagem de estados
1.1 Diagrama de transição de estado
1.1.1 Estados
1.1.2 Transições
1.1.3 Eventos
1.1.4 Condição de guarda
1.1.5 Ações
1.1.6 Atividades
1.1.7 Ponto de junção
1.1.8 Cláusulas entry, exit e do
1.1.9 Transições internas
1.1.10 Estados aninhados
1.1.11 Estados concorrentes

Trabalho(Valor = 3,0 Pontos)

Realizar trabalho de pesquisa sobre:
2 - Modelagem de atividades
2.1 Diagrama de atividade
2.1.1 Fluxo de controle sequencial
2.1.2 Fluxo de controle paralelo
2.1.3 Raias de natação

Livro: Bezerra, Eduardo, 1972- Princípios de análise e projeto de sistemas com UML / Eduardo Bezerra. - [3. ed.]

Bibliografia Base

Eduardo Bezerra - Princípios de Análise e Projeto de Sistema com UML. Rio de Janeiro: Elsevier Editora Ltda, 2007.

Antonio Lopes Marinho – Análise e modelagem de sistemas. São Paulo: Pearson Education do Brasil, 2016.

Roger S. Pressman – Engenharia de software : uma abordagem profissional. 7ª Edição. Porto Alegre: AMGH Editora Ltda, 2011.

Ian Sommerville – Engenharia de Software. 9ª Edição. São Paulo: Pearson Education do Brasil, 2011

Perguntas?

Conclusão