

Unidade 03

Diagrama de Classe, Sequência e Comunicação



Introdução da Unidade

Introdução

Olá, amigo(a) discente! Seja bem-vindo(a)!

Nesta unidade, vamos dar continuidade e finalizar o diagrama de classe. Além disso, vamos aprender a trabalhar com o diagrama de interação que é composto pelos diagramas de Sequência e Comunicação. Para a elaboração do diagrama de sequência é necessário ter elaborado os diagramas de caso de uso e classe, só assim será possível elaborar o diagrama de sequência do caso de uso *sucLocarAutomóveis* referente ao estudo de caso de Locação de Veículos.

Objetivos

- Aprender a trabalhar com associações no diagrama de classe;
- Modelar um diagrama de classe;
- Entender os conceitos do diagrama de interação;
- Modelar um diagrama de sequência e comunicação de um caso de uso;
- Rastreabilidade do diagrama de caso de uso para o diagrama de classe e para o diagrama de sequência/comunicação.

Conteúdo Programático

Aula 01 – Diagrama de Classe (continuidade).

Aula 02 – Diagrama de Sequência.



Quer **assistir às videoaulas** em seu celular? Basta apontar a câmera para os **QR Codes** distribuídos neste conteúdo.

Caso necessário, instale um aplicativo de leitura QR Code no celular e efetue o login na sua conta Gmail.

DIAGRAMA DE CLASSE (continuidade)

Padrões para nomenclatura de classes

Quando vamos atribuir um nome a uma determinada classe alguns padrões devem ser seguidos, uns por restrições das linguagens de programação, outros por motivos de qualidade e outros ainda adquiridos pela experiência em desenvolvimento de *software* passada de “geração em geração” por desenvolvedores, abaixo serão citadas algumas regras:

- Todo o nome de classe deve começar com uma letra em caixa alta (maiúscula) seguido do restante das letras em caixa baixa (minúsculas);
- O nome da classe deve estar relacionado ao tipo de informação que ela manipula;
- Quando houver nomes de classes, como “ser humano”, a regra é a seguinte:
 - Cada nome é iniciado com letra maiúscula seguido das letras minúsculas;
 - Por padrão, nomes compostos formam uma única palavra;
 - Exemplos de classes com nomes compostos:
 - SerHuman;
 - AntVirus;
 - HotDog.
- Nomes de classes não podem começar com números (porém podem conter números);
- Por uma questão de organização, não é aconselhável acentuar nomes de classes; a linguagem Java, por exemplo, não acusará nenhum erro, e irá compilar a classe gerando o arquivo com extensão *class*.

Padrões para nomenclatura de métodos e atributos

Com relação a padronização para nomenclatura de métodos e atributos de classe ou variáveis, vale ressaltar a regra das letras minúsculas e maiúsculas. A assinatura de um método se difere da assinatura de uma classe em alguns aspectos:

- Em primeiro lugar, nomes de métodos devem começar com letras minúsculas;
- Os verbos utilizados para nomear os métodos devem estar no imperativo (ar, er, ir);
- Assim como os nomes de classes, nomes de métodos devem indicar comportamentos executados pelos próprios métodos;
- Para nomes compostos, utiliza-se a mesma regra de nomenclatura de classes.

Estereótipo

O estereótipo indica como uma classe é utilizada pelo projeto. Ele não faz parte do nome da classe, e possui sua definição entre dois sinais de menor (<<) e maior (>>). Eles são mecanismos de extensibilidade da UML que permitem classificar elementos que possuem

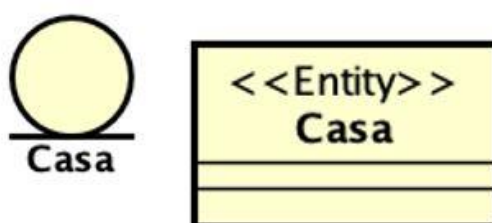
semelhanças. Os símbolos utilizados para sua representação geralmente são colocados após o elemento que está sendo “estereotipado”, e em seu interior está a descrição atribuída ao elemento.

Existem 3 estereótipos bastante utilizados nos diagramas de classe da UML, sendo <<entity>>, <<boundary>> e <<control>>.

- **Estereótipo << entity >>**

Este estereótipo tem como principal objetivo deixar claro que a classe é uma classe de entidade, ou seja, as classes contêm informações recebidas e armazenadas pelo sistema ou geradas por meio desse (GUEDES, 2009). A Figura 1 exibe a entidade **Casa** que pode ser também representada com o estereótipo <<entity>> e assumir duas formas de visualização.

Figura 1 - Exemplo << entity >>



Fonte: o autor (2021)

- **Estereótipo << boundary >>**

Este estereótipo é conhecido como fronteira, identifica que uma classe serve de comunicação entre atores externos e o sistema. Muitas vezes necessita interagir com outra classe do tipo <<control>> (GUEDES, 2009).

- **Estereótipo << control >>**

Estes estereótipos são responsáveis por interpretar eventos ocorridos sobre os objetos <<boundary>>, como, por exemplo o movimento do mouse ou pressionar de um botão e retransmiti-lo aos objetos de uma classe de entidade que compõem o sistema. (GUEDES, 2009).

A Figura 2 demonstra a Interface_Banco (**boundary**), e seus objetos como caixa de seleção, botões etc. Os eventos que ocorrem nesses objetos são repassados para o objeto da classe Controlador_Banco (**control**) que repassará a outros objetos do sistema, aqui ainda não representados.

Figura 2 - Exemplo de << boundary >> e << control >>



Fonte: o autor (2021)

Visibilidade das classes

A visibilidade das classes, assim como a dos atributos, indica o nível de acesso delas em relação aos outros componentes ou classes que compõem o projeto de *software* ao qual essa classe pertença, são eles:

- **Public: (+)** a visibilidade pública torna a classe acessível a todas as classes e componentes do sistema, e seus atributos e métodos podem ser acessados por meio da instância de um objeto desta classe.
- **protected: (#)** a visibilidade protegida torna os atributos e métodos da classe visíveis e utilizáveis somente por subclasses da classe protegida, assim como a própria classe. Atributos e métodos dessa classe só podem ser acessados por subclasses ou pela própria classe que usa esse modificador de acesso.
- **private: (-)** o tipo de acesso privado restringe o acesso aos atributos e métodos da classe somente à própria classe. (Visível apenas dentro do *namespace* que possui).



Videoaula 1

Utilize o QR Code para assistir!

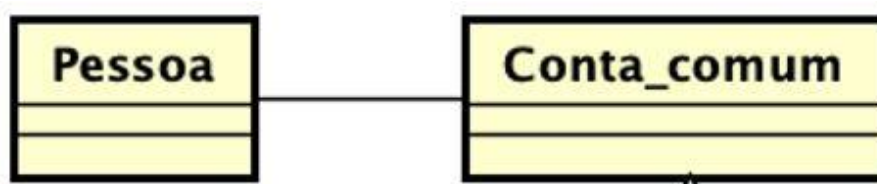
Neste momento, vamos assistir ao vídeo sobre o Diagrama de Classe.



Associação

Relacionamento normal (binária): às vezes é necessário estabelecer um relacionamento entre classes, para que uma saiba os comportamentos da outra. A Figura 3 apresenta duas classes com o relacionamento binário. Este tipo de relacionamento são os mais comuns no diagrama de classe.

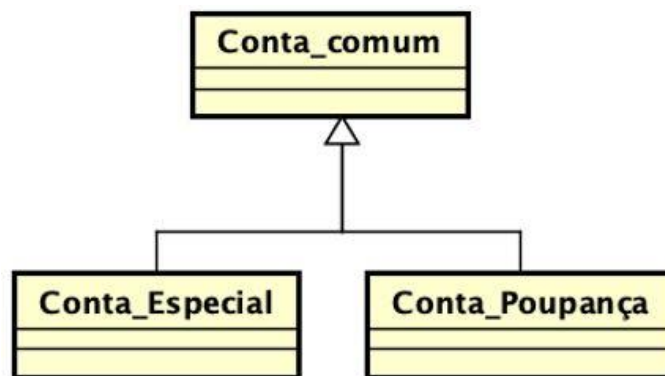
Figura 3 - Associação Binária



Fonte: o autor (2021)

Generalização: é a capacidade de uma classe ou objeto herdar todas as características de uma super classe ou classe pai, com isso, essa classe, chamada sub-classe torna-se uma “super-classe mais especializada”, pois possui características e comportamentos herdados de sua classe pai, e ainda possui as características e comportamentos específicos dessa classe. A notação de generalização é a mesma utilizada no diagrama de caso de uso. A Figura 4 demonstra um exemplo de generalização, na qual a classe **Conta_comum** é a classe pai e as classes **Conta_Especial** e **Conta_Poupanca** são as classes filhas, que herdam as características da classe pai, como atributos e métodos (não demonstrados nessa imagem).

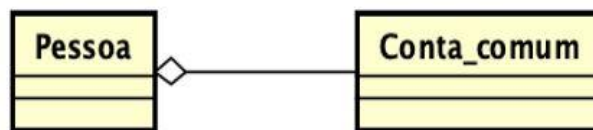
Figura 4 - Exemplo de Generalização



Fonte: o autor (2021)

Agregação: é um tipo de associação entre classes no qual relacionamos o todo com suas partes. Esse tipo de associação também é conhecido como relação de conteúdo, e indica que uma das classes contidas no relacionamento é uma parte ou está contida em outras classes. A Figura 5 apresenta um exemplo de agregação no qual o objeto **Pessoa** é objeto-todo e a **Conta_comum** o objeto-parte. Isso demonstra que quando uma pessoa for consultada, junto com a consulta da pessoa serão demonstradas todas as contas que ele possui.

Figura 5 - Exemplo de Agregação



Fonte: o autor (2021)

Composição: é um tipo de uma agregação onde uma classe “vive” e compõe a outra. E caso haja a destruição do objeto da classe que contém, as demais classes da agregação serão destruídas também, pois fazem parte uns dos outros. Na Figura 6 é demonstrado um exemplo de composição. No caso do objeto **Revista**, se for deletada, a **Edição** será automaticamente destruída. Ou seja, as partes não podem sobreviver ao conjunto/agregado.

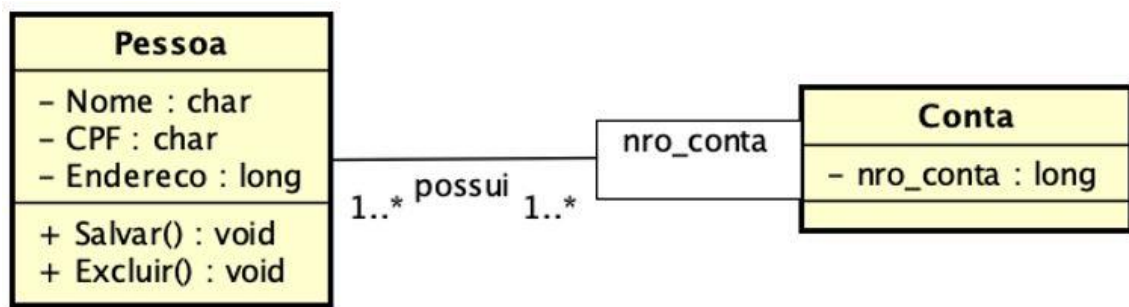
Figura 6 - Exemplo de Composição



Fonte: o autor (2021)

Qualificador: Associações qualificadas são usadas com associações de um para vários (1..*) ou vários para vários (*). O “qualificador” (identificador da associação qualificada) especifica como um determinado objeto no final da associação “n” é identificado, e pode ser visto como um tipo de chave para separar todos os objetos na associação. O identificador é desenhado como uma pequena caixa no final da associação junto à classe de onde a navegação deve ser feita. (Figura 7)

Figura 7 - Exemplo de Qualificador



Fonte: o autor (2021)



Videoaula 2

Utilize o QR Code para assistir!

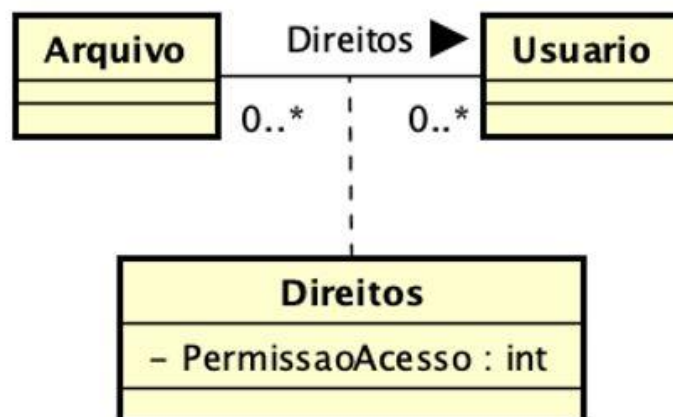
Neste momento, vamos assistir ao vídeo sobre o Diagrama de Classe.



Tipos de classe

Associativa: é uma classe interligada a uma associação ao invés de estar ligada a outras classes (se comparada ao banco de dados, indicaria o relacionamento N x N). A Figura 8 mostra um exemplo de classe associativa.

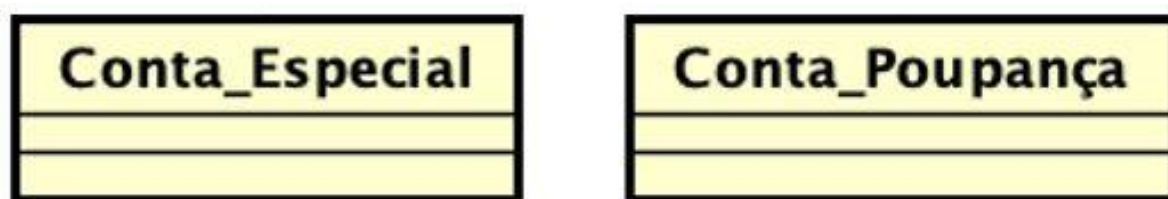
Figura 8 - Exemplo de Classe Associativa



Fonte: o autor (2021)

Classe concreta: também chamada de classe, é uma classe que possui métodos e atributos e pode ser utilizada por outras classes do sistema. A Figura 9 apresenta um exemplo de uma classe concreta com seus atributos e métodos. Classes concretas são classes que foram instanciadas.

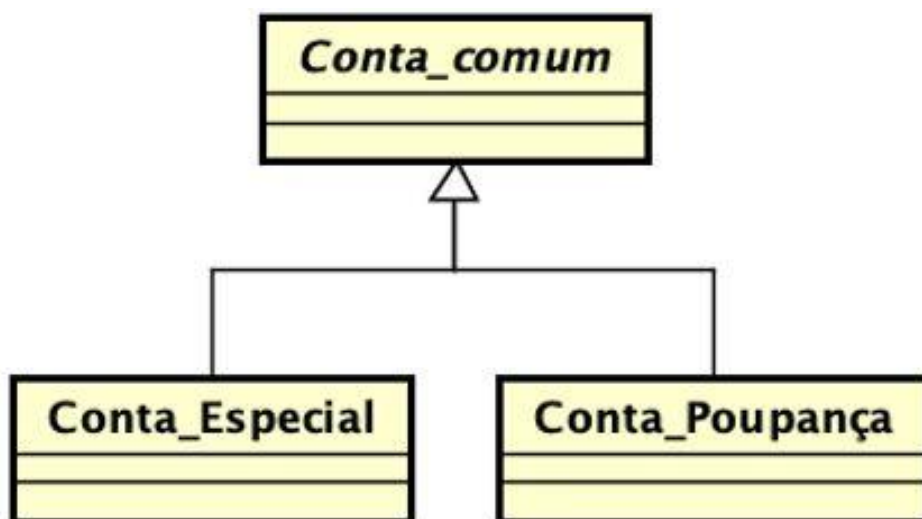
Figura 9 - Exemplo de Classe Concreta



Fonte: o autor (2021)

Classe abstrata: são classes mais “especializadas” que possibilitam herança de suas características por outras classes do sistema. Objetos desse tipo de classe não podem ser instanciados. Uma classe abstrata é representada em itálico *Conta_Comum* conforme apresentado na Figura 10.

Figura 10 - Exemplo de Classe abstrata

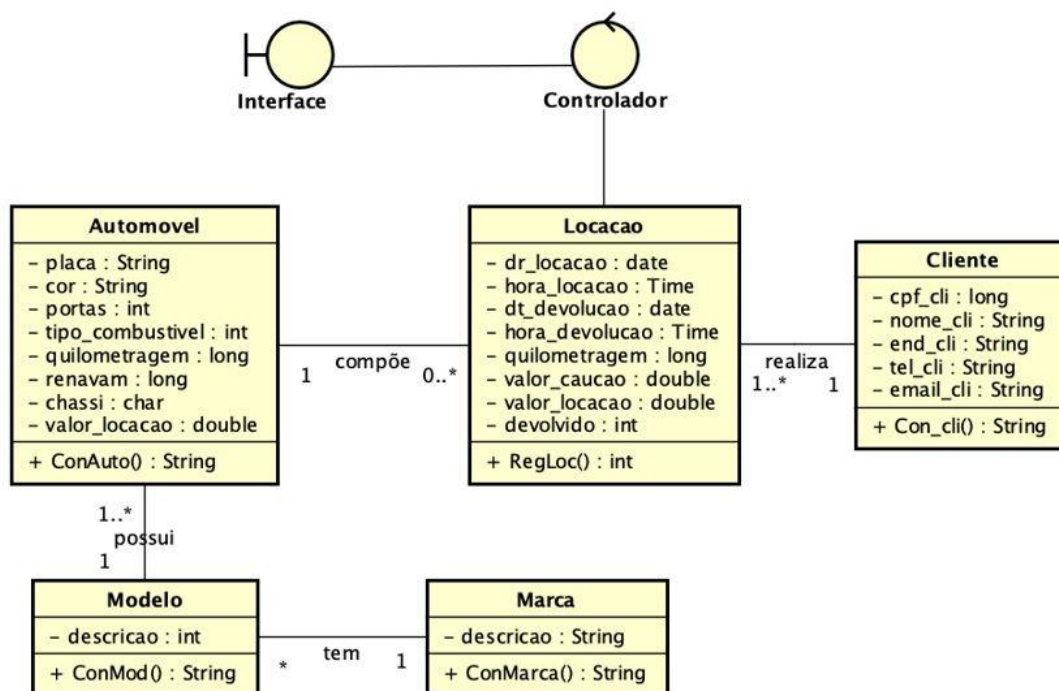


Fonte: o autor (2021)

Diagrama de Classe do Sistema de Locação de Veículos

A Figura 11 apresenta o diagrama de classe do estudo de caso do Sistema de Locação de Veículos (GUEDES, 2009).

Figura 11 - Exemplo do Diagrama de Classe



Fonte: Guedes (2009)

Marca

Essa classe representa as marcas dos carros disponíveis na locadora. Tem como atributo a descrição da marca, do tipo *String*, além do método *ConMarca*, usado para consultar uma marca específica. Como pode-se perceber, o método retorna uma *String*;

Modelo

Já essa classe armazena os modelos das marcas disponíveis na locadora. A semelhança da classe *Marca*, essa classe tem como atributo a descrição do modelo e um método que permite consultar um determinado modelo. Observe que um modelo tem somente uma marca, mas uma marca pode ter muitos modelos;

Cliente

Essa classe, por sua vez, armazena as informações relativas aos clientes que locam veículos. Seus atributos são autoexplicativos. A classe tem ainda um método para consultar um determinado cliente e retornar seus dados.

Automóvel

Essa classe contém as informações dos automóveis locados pela empresa, indo desde a placa do veículo até o valor da locação. Existe, ainda, um método que permite consultar um determinado automóvel, retornando os dados deste. Um automóvel está relacionado a um único modelo, mas um modelo pode estar associado a muitos automóveis.

Locação

Essa classe representa as informações relativas às locações de automóveis já realizadas. A classe tem os atributos data de locação e data de devolução, do tipo *Date*; hora de locação e

hora de devolução do tipo *Time*; quilometragem, do tipo *long*; valor de caução e de locação, do tipo *Double*; e um atributo chamado "devolvido", do tipo *int*; cuja função é determinar se a locação foi devolvida ou não. A classe tem ainda um método para registrar uma locação, que retorna verdadeiro se for executado com sucesso e falso em caso contrário. Ao examinarmos as associações dessa classe, concluímos que um cliente pode realizar muitas locações (no mínimo uma), mas que uma locação se refere a um único cliente, e que uma locação se refere a um veículo em particular, mas que um mesmo veículo pode estar relacionado a muitas locações ou não ter sido locado nunca.



Videoaula 3

Utilize o QR Code para assistir!

Neste momento, vamos assistir ao vídeo sobre o Diagrama de Classe.





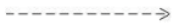



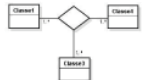



Leitura Obrigatória

Leia o capítulo 6 do livro do Medeiros. *Desenvolvendo Software com UML 2.0*.

Acesse o livro no *link* abaixo:

Disponível em: <https://plataforma.bvirtual.com.br/Leitor/Publicacao/2921/pdf/0>. Acesso em: 5 set. 2021.

Quadro 1 - Notação do Diagrama de Classe

Nome			Definição	
Classe			Representa uma categoria, e os objetos são os membros ou exemplos dessa categoria. Em geral, uma classe tem atributos e métodos, mas é possível encontrar classes que contenham apenas uma dessas características ou mesmo nenhuma delas, como no caso de classes abstratas. Tais classes podem ser de dois tipos: abstratas e concretas.	
Multiplicidade			Especifica a quantidade de valores que podem estar associados a um elemento do modelo.	[0..5] ou 0..5
Relacionamento			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 (GUEDES, 2009)	
Relacionamento	Realização (Interface)		É um tipo de relacionamento especial que mistura características dos relacionamentos de generalização e dependência, sendo usada para identificar classes responsáveis por executar funções para outras classes (GUEDES, 2009)	
	Dependência		É o relacionamento, como o próprio nome diz, identifica certo grau de dependência de uma classe em relação à outra (GUEDES, 2009).	
	Generalização		O objetivo dessa associação é representar a ocorrência de herança entre as classes, identificando as superclasses, chamadas gerais e subclasses, chamadas especializadas, demonstrando a hierarquia entre as classes e possivelmente métodos polimórficos nas classes especializadas (GUEDES, 2009).	
	Associação		Uma associação descreve um vínculo que ocorre normalmente entre os objetos de uma ou mais classes (GUEDES, 2009).	
Associação		Unidirecional	Relacionamento de um objeto de uma classe com objetos da mesma classe (GUEDES, 2009).	
		Bidirecional	Relacionamento entre objetos de duas classes distintas (GUEDES, 2009).	
	Ternária		São associações que conectam objetos de mais de duas classes. São representadas por um losango para onde convergem todas as ligações da associação (GUEDES, 2009).	
	Agregação		É um tipo especial de associação onde se tenta demonstrar que as informações de um objeto (chamado objeto-todo) precisam ser complementadas pelas informações contidas em um ou mais objetos de outra classe (chamados objetos-parce). Esse tipo de associação tenta demonstrar uma relação todo/parte entre os objetos associados (GUEDES, 2009).	
	Composição		É um tipo de composição da qual constitui-se em uma variação da agregação, onde é apresentado um vínculo mais forte entre os objetos-todo e os objetos-parce, procurando demonstrar que os objetos-parce têm de estar associados a um único objeto-todo(GUEDES, 2009).	
	Classe Associativa		São aquelas produzidas quando da ocorrência de associações que tenham multiplicidade muitos (*) em todas as suas extremidades (GUEDES, 2009).	
Estereótipo			Permite a extensibilidade aos componentes ou associação da UML.	<<entity>> <<boundary>> <<control>> <<enumeration>>
Atributos			Representam as características de uma classe, ou seja, as peculiaridades que costumam variar de um objeto para outro (GUEDES, 2009).	
Atributo	Derivado		É o atributo do qual seus valores são representados por algum tipo de cálculo. Nestes casos, o atributo é representado por uma / na frente do atributo (GUEDES, 2009).	/atributo
	Estático		São atributos cujos valores são idênticos para todos os objetos de uma classe, ou seja, é um atributo pertencente à classe propriamente dita. Sua identificação se dá por meio do nome da variável sublinhado (GUEDES, 2009).	<u>atributo</u>
	Tipos		O tipo de atributo identifica um classificador que explica a espécie de informação que pode ser armazenada no atributo (PENDER, 2004).	int, boolean, varchar, double
Visibilidade			É utilizada para indicar o nível de acessibilidade de um determinado atributo ou método, sendo representada à esquerda destes, existindo basicamente quatro modos de visibilidade: público, protegido, privado e pacote (GUEDES, 2009).	
Visibilidade	Pública		Determina que o atributo ou método pode ser utilizado por qualquer objeto (GUEDES, 2009).	+ atributo3
	Privada		Somente os objetos da classe detentora do atributo ou método poderão enxergá-lo (GUEDES, 2009).	- atributo1
	Pacote		Determina que o atributo ou método é visível por qualquer objeto dentro do pacote. Somente elementos que fazem parte de um pacote podem ter essa visibilidade. Nenhum elemento fora do pacote poderá ter acesso a um atributo ou método com essa visibilidade.	~ atributo4
	Protegida		Determina que além dos objetos da classe detentora do atributo ou método também os objetos de suas subclasses poderão ter acesso ao mesmo (GUEDES, 2009).	# atributo2
Operação			Também conhecido como método ou comportamento da qual representa uma atividade que um objeto de uma classe pode executar (GUEDES, 2009).	operação() : int operação() : boolean

Fonte: o autor (2021)

DIAGRAMA DE SEQUÊNCIA

Diagrama de Sequência

Este é um diagrama comportamental que procura determinar a sequência de eventos que ocorrem em um determinado processo, identificando quais mensagens devem ser disparadas entre os elementos envolvidos e em que ordem. Assim, determinar a ordem em que os eventos ocorrem, as mensagens que são enviadas aos métodos que são chamados e como os objetos interagem dentro de um determinado processo é o objetivo principal deste diagrama (Guedes, 2009).

Normalmente temos um diagrama da sequência para cada caso de uso. O diagrama de sequência depende do diagrama de classe, uma vez que a classe que aparece no diagrama de sequência são as mesmas do diagrama de classes.

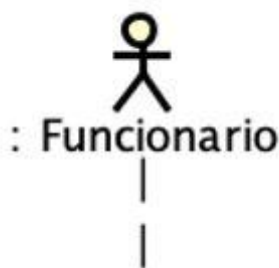
Segundo o RUP (2006), a UML 2.0 possui vários novos recursos para diagramas de sequência:

Segue abaixo os principais conceitos utilizados no diagrama de sequência.

Ator

Representa os papéis desempenhados pelos diversos usuários que poderão utilizar, de alguma maneira, os serviços e funções do sistema. Os atores, neste diagrama, são instâncias dos atores declarados no diagrama de Casos de Uso, representam entidades externas que iniciam processos. No diagrama de sequência o ator é representado de acordo com a Figura 12.

Figura 12 - Ator



Fonte: o autor (2021)

Objeto/Classe

Objetos são membros ou exemplos de uma determinada categoria, que é representada por uma classe. Na Figura 19 é possível visualizar o exemplo de objetos.

Linha de Vida

A linha de vida representa o tempo em que um objeto existe durante um processo. As linhas de vida são representadas por linhas finas verticais tracejadas, partindo do retângulo que representa o objeto (PENDER, 2004). Na Figura 19 é possível visualizar o exemplo de linha de vida.

Foco de Controle

Indica os períodos em que um determinado objeto está participando ativamente do processo, ou seja, identifica os momentos em que um objeto está executando um ou mais métodos utilizados em um processo específico. Os focos de controle são representados dentro da linha de vida de um objeto (GUEDES, 2009). Na Figura 19 é possível visualizar o exemplo de foco de controle.



Videoaula 1

Utilize o QR Code para assistir!

Agora, assista o vídeo sobre o Diagrama de Sequência.

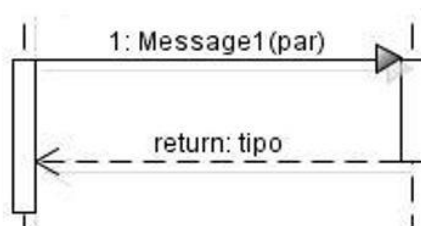


Mensagem

As mensagens são utilizadas para demonstrar a ocorrência de eventos, que normalmente forçam a chamada de um método em alguns dos objetos envolvidos no processo. Pode ocorrer, no entanto, de uma mensagem representar a comunicação entre dois atores, nesse caso, não disparando métodos (GUEDES, 2009). Na Figura 19, é possível visualizar o exemplo de mensagem.

- **Mensagem Síncrona:** Uma mensagem síncrona considera que um retorno é necessário, de modo que o transmissor espera pelo retorno antes de prosseguir com qualquer outra atividade. Veja o exemplo na Figura 13.

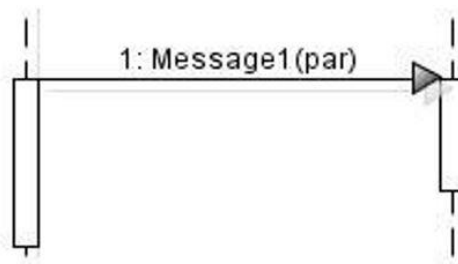
Figura 13 - Mensagem Síncrona



Fonte: o autor (2021)

- **Mensagem Assíncrona:** Uma mensagem assíncrona diz algo sobre as responsabilidades do transmissor e do receptor. O emissor é responsável apenas por levar a mensagem ao receptor. Veja o exemplo na Figura 14.

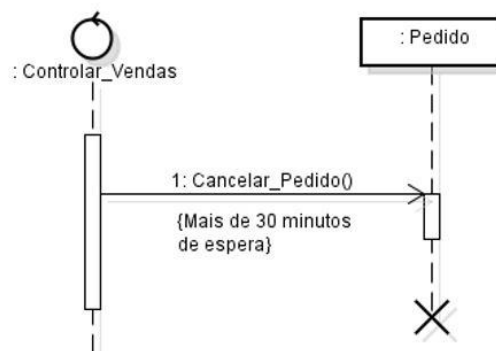
Figura 14 - Mensagem Assíncrona



Fonte: o autor (2021)

- **Timeout:** Quando uma mensagem é cancelada dependendo do tempo de duração. Veja o exemplo na Figura 15.

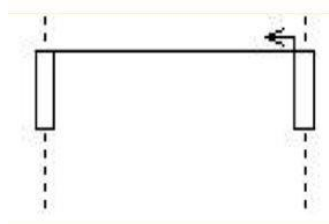
Figura 15 - Mensagem Timeout



Fonte: o autor (2021)

- **Balking:** Se o destinatário da mensagem não estiver imediatamente pronto para aceitar a mensagem, o remetente aborta a mensagem e continua o processamento. Veja o exemplo na Figura 16.

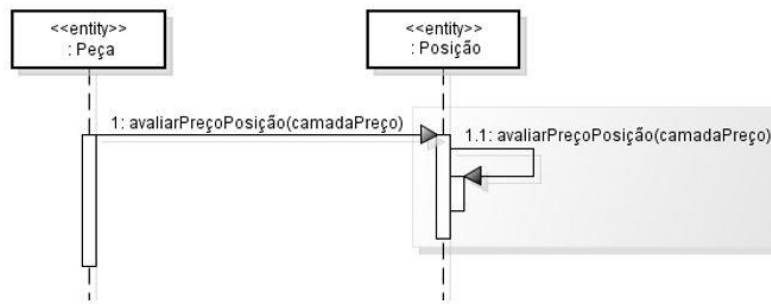
Figura 16 - Mensagem Balking



Fonte: o autor (2021)

- **Recursão:** Um objeto também poderia ter que chamar uma mensagem recursivamente, ou seja, chamar a mesma mensagem de dentro da mensagem. Veja o exemplo na Figura 17.

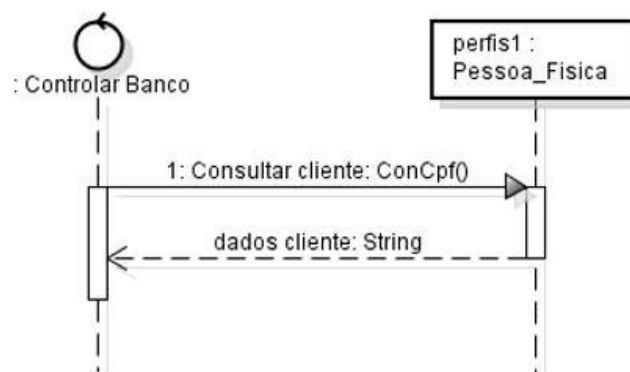
Figura 17 - Exemplo de Recursão



Fonte: o autor (2021)

- **Retorno:** Esse tipo de mensagem identifica a resposta a uma mensagem para o objeto ou ator que a chamou. Uma mensagem de retorno pode retornar informações específicas do método chamado ou apenas um valor indicando se o método foi executado com sucesso ou não. Veja o exemplo na Figura 18.

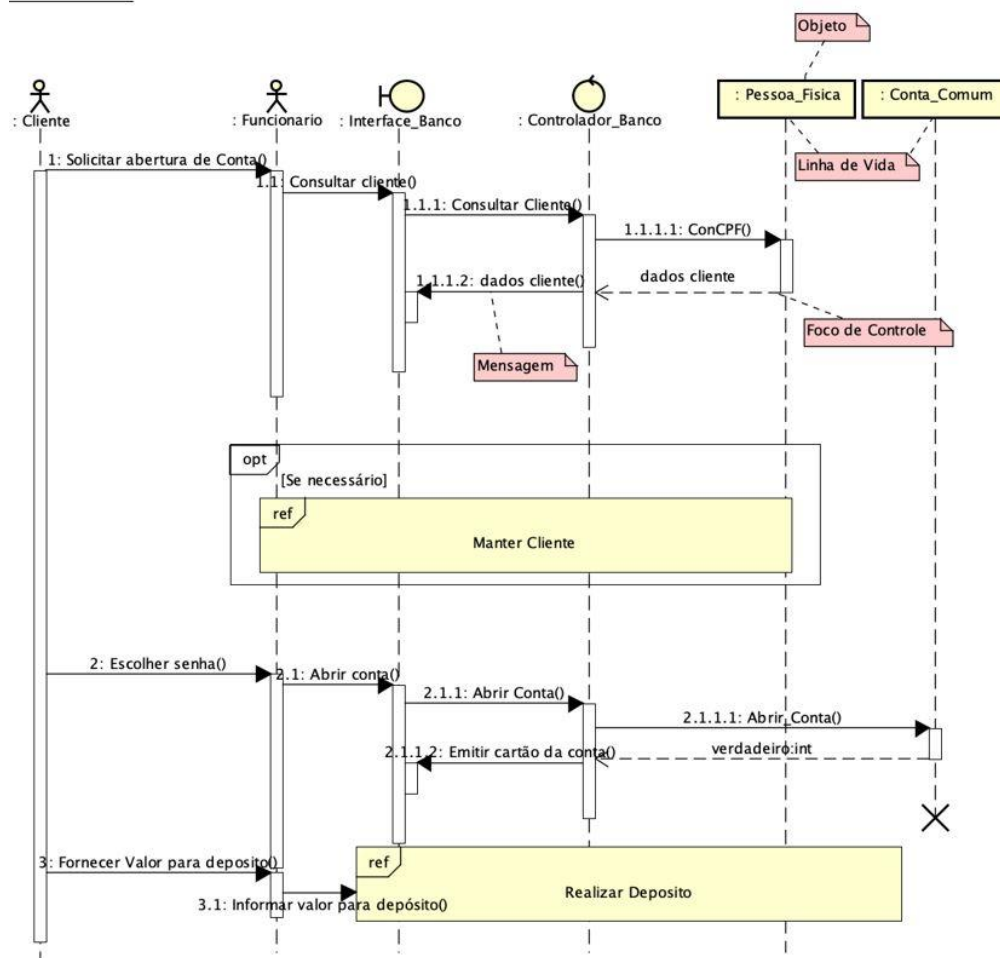
Figura 18 - Mensagem Retorno



Fonte: o autor (2021)

Segue abaixo a figura 19, com os exemplos dos conceitos explicados acima. (GUEDES, 2009).

Figura 19 - Exemplo do Diagrama de Sequência



Fonte: Guedes (2009)



Videoaula 2

Utilize o QR Code para assistir!

Agora, assista o vídeo sobre o Diagrama de Sequência.



Fragmentos de Interações

Fragmentos fornecem uma semântica mais clara de como ocorre o comportamento em um diagrama de sequência. Um fragmento combinado encapsula partes de um diagrama de sequência, no qual fluxos separados podem ser modelados, mostrando como as condições levam a caminhos alternativos de execução.

As ocorrências de interação permitem a decomposição das interações em amostras reutilizáveis. É uma maneira útil de compartilhar partes de uma interação entre várias outras interações.

Na UML 1.x, uma representação possível para *loops* era utilizar a condição de *loop* escrita em uma Nota. A Nota era anexada à mensagem ou ao conjunto de mensagens a serem executadas, enquanto a condição de *loop* fosse verdadeira. Na UML 2.0, há uma representação específica para *loops*. Na UML 2.0, diagramas de sequência podem mostrar como os objetos são criados e eliminados.

Execução de Ocorrência mostra o foco de controle que um objeto executa em um determinado momento, quando recebe uma mensagem. Com os novos recursos para representar fragmentos, ocorrências de interação e *loops*, os diagramas de sequência podem ser utilizados de duas formas:

Forma de instância:

Descreve um cenário específico detalhadamente, documentando uma interação possível, sem condições, ramificações ou *loops*. Essa forma é utilizada para representar um cenário de caso de uso. Cenários diferentes do mesmo caso de uso são representados em diagramas de sequência diferentes. As ferramentas de modelagem que suportam a semântica da UML 1.x só permitem essa forma de representação.

Forma genérica:

descreve todas as alternativas possíveis de um cenário, aproveitando as vantagens dos novos recursos da UML 2.0, como condições, ramificações e *loops*. Essa forma pode ser utilizada para representar diversos cenários do mesmo caso de uso em um diagrama de sequência exclusivo, no qual faça sentido.

A Figura 20, a seguir, mostra um exemplo de diagrama de sequência que modela cenários diferentes.

Segue abaixo alguns dos fragmentos utilizados:

Alt - Alternativo. Define que o fragmento representa uma escolha entre dois ou mais comportamento. Normalmente, utiliza-se a condição de guarda para definir a escolha do comportamento. Se analisarmos a Figura 10, se o **Saldo positivo** irá executar o processo Realizar Saque e se o **Saldo Negativo**, deverá executar o processo para Realizar Depósito.

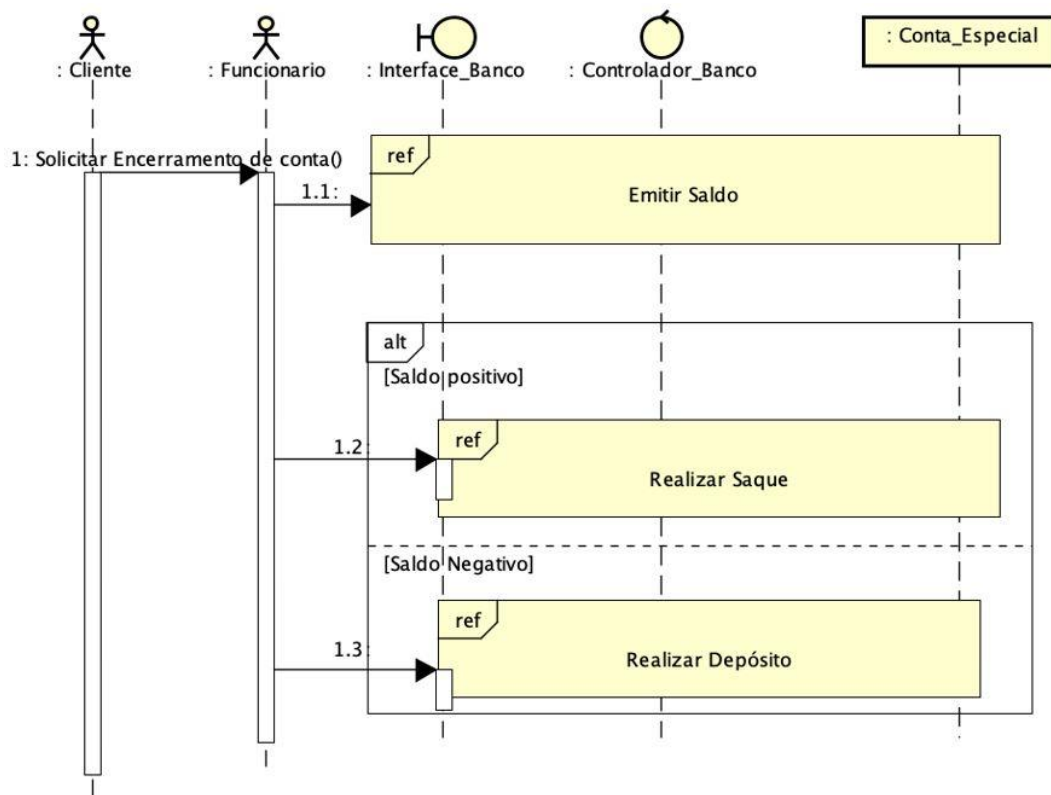
Opt - Opção. Determina se os passos do fragmento serão ou não executados, dependendo da condição atendida. De acordo com a Figura 9, só será executado o processo **Manter Cliente**, "Se Necessário".

Ref - Referido - Este operador indica que o se deve procurar por um diagrama com o nome apresentado dentro do fragmento Ref. De acordo com a Figura 10, é apresentado o "**Realizar Depósito**" dentro do Ref, isso significa que esse diagrama está fazendo referência a outro diagrama.

Loop - Laço - Este operador representa que o conteúdo do fragmento pode ser repetido por diversas vezes, dependendo da condição atribuída. Na Figura 11, o *loop* acontecerá enquanto tiver cliente.

A Figura 20 exemplifica a utilização de variados tipos de fragmentos, sendo alguns deles utilizados de forma combinada, como o **alt** e **ref**.

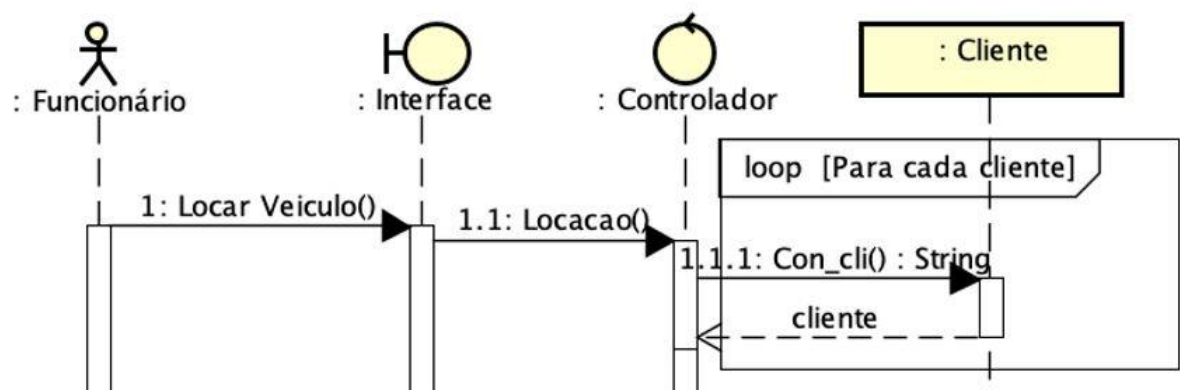
Figura 20 - Exemplo de Fragmentos do Diagrama de Sequência



Fonte: Guedes (2009)

Na Figura 21, pode-se observar uma parte do diagrama de sequência, onde é possível identificar o uso do fragmento *loop*.

Figura 21 - Exemplo de Fragmento Loop



Fonte: Guedes (2009)



Videoaula 3

Utilize o QR Code para assistir!

Agora, assista o vídeo sobre os Fragmentos do Diagrama de Sequência.



DIAGRAMA DE COMUNICAÇÃO/COLABORAÇÃO

Na UML 1.5, este diagrama era chamado de colaboração. Na UML 2.0 o nome foi alterado para diagrama de comunicação. Ele mostra a interação entre as classes. As classes colaboram enviando mensagens umas para as outras. São objetos, instanciados na memória que enviam mensagens entre um objeto e outro.

Este diagrama mostra os objetos ligados por uma linha sólida, identificada por uma mensagem e com números mostrando a sequência em que essas mensagens devem ser lidas, pensadas e codificadas.

Concluindo, esse diagrama, comunicação, faz parte dos chamados diagrama de interação. Pode haver trechos onde, em um diagrama de classes, uma colaboração pode ser representada a fim de prover melhor esclarecimento.

A utilização desse diagrama é opcional e pode ser realizada em qualquer momento da modelagem. Pode ser útil, em colaborações complexas, mas não supre as finalidades de modelagem como o diagrama de sequência. Se colocarmos, no diagrama de sequência, a numeração das mensagens, o diagrama de comunicação fica prejudicado no seu uso. O Quadro 1, apresenta a diferença em um comparativo do diagrama de sequência *versus* o diagrama de comunicação.

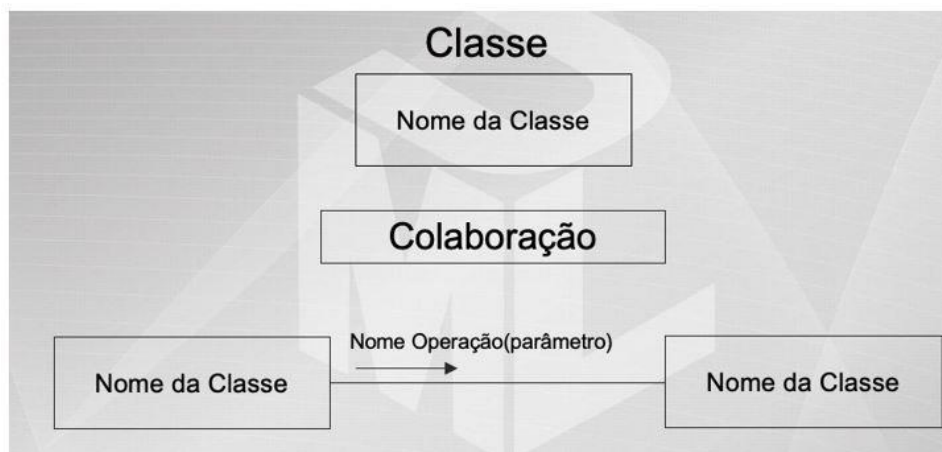
Quadro 1 - Comparação do diagrama de sequência versus o diagrama de comunicação

Sequência x Comunicação	
SEQUÊNCIA	COLABORAÇÃO
<ul style="list-style-type: none">• Ênfase à ordenação temporal da mensagens;• Graficamente, um diagrama de Sequência é uma tabeça qe mostra objetos distribuídos no eixo X e mensagens, em ordem crescente no tempo, no eixo Y;• Possui uma linha de vida do objeto;• Existe o Foco de Controle (mostra o período durante o qual um objeto está desempenhando uma ação.	<ul style="list-style-type: none">• Ênfase aos relacionamentos estruturais existente entre os objetos que interagem;• Graficamente, um diagrama de Colaboração é uma coleção de vértices e arcos;• Possui o caminho (local, global);• Existe o número de sequência.

Fonte: o autor (2021)

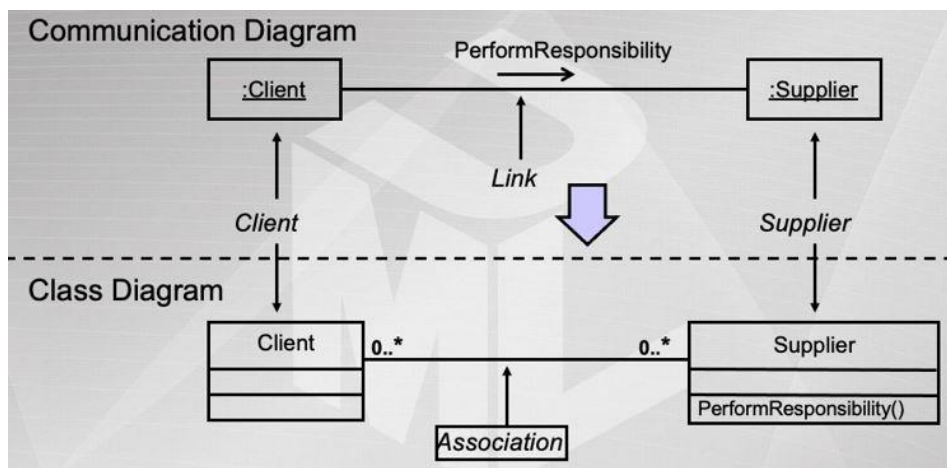
As Figuras 22 e 23 mostram dois exemplos em relação a sintaxe quanto ao uso do diagrama de comunicação.

Figura 22 - Classes versus Colaboração



Fonte: o autor (2021)

Figura 23 - Comunicação versus o diagrama de classes



Fonte: o autor (2021)

Leitura Obrigatória

Leia o capítulo 1, 2, e 3 do livro do Medeiros. *Desenvolvendo Software com UML 2.0*.

Acesse o livro no *link* abaixo:

Disponível em: <https://plataforma.bvirtual.com.br/Leitor/Publicacao/2921/pdf/0>. Acesso em: 5 set. 2021.

Leitura Obrigatória

Leia o capítulo 8 e 14 do livro do Medeiros. *Desenvolvendo Software com UML 2.0*.

Acesse no *link* abaixo:

Disponível em: <https://plataforma.bvirtual.com.br/Leitor/Publicacao/2921/pdf/0>. Acesso em: 5 set. 2021.

Videoaulas

Prezado(a) aluno(a)!

Você também poderá encontrar todas as videoaulas, clicando em “**Módulos**” no “**Menu Lateral**” e acessar a página de vídeos.

Encerramento

Nesta unidade, foram abordados o Diagrama de Sequência e Diagrama de Colaboração.

Na nossa próxima unidade, vamos dar continuidade ao Diagrama de Sequência e muitos outros conteúdos. Até a próxima unidade.

Referências

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML – Guia do Usuário**. 2. ed. Rio de Janeiro: Editora Campus, 2006.

FOWLER, M. **UML Essencial**. 3. ed. Porto Alegre: Bookman, 2007.

GUEDES, G. T. A. **UML 2.0: uma abordagem prática**. São Paulo: Novatec, 2009.

IBM. RUP – Rational Unified Process (Software) Versão 7.0. USA: IBM Rational, 2006.

LARMAN, C. **Utilizando UML e Padrões**. 3. ed. Porto Alegre: Bookman, 2007.

MEDEIROS, E. **Desenvolvendo Software com UML 2.0**. São Paulo: Ed. Pearson, 2004.

OMG. OBJECT MANAGEMENT GROUP. UML 2.0. Disponível em: www.omg.org. 2020.

PAGE-JONES, Meilier. **Fundamentos do Desenho Orientado a objetos com UML**. São Paulo: Makron books, 2001.

PENDER, T. **UML 2.0 - A Bíblia**. Rio de Janeiro: Editora Campus, 2004.

TANAKA, S. S. **O poder da tecnologia de workflow e dos mapas conceituais no processo de ensino e aprendizagem da UML**. Dissertação de Mestrado, UEL. Londrina, 2011.



UNIFIL.BR