

7-Diagramas UML (DL)

E-Book - Apostila

Introdução à UML

Vamos iniciar nosso conteúdo sobre UML!

'No final dos anos 80 e início dos anos 90, tínhamos muitos conflitos de definições e nomenclaturas na área de modelagem. A escolha para utilização de um determinado padrão era definido mais pelo “gosto” pessoal do que por fatores técnicos oferecidos. Então, os três mais respeitados nomes nesse campo, cada qual com seu conceito e implementação de modelo, Ivar Jacobson (OOSE – Object Oriented Software Engineering), Grady Booch (The Booch Method) and James Rumbaugh (OMT – Object Modeling Technique) decidiram por fim aos debates e trabalhar juntos na definição de um modelo único que veio a ser a UML.' (INTRODUÇÃO..., 2009, on-line).

A UML permite que você “desenhe” uma “planta” do seu sistema. Podemos comparar com um engenheiro que vai realizar um projeto sem, antes, ter completado toda a planta que define o desenho da estrutura a ser construída. A experiência do engenheiro garante, até certo ponto, o caminhar do projeto, contudo, se houvesse a planta, os cálculos estruturais e todo o planejamento suportado pelos documentos corretos, o sucesso do projeto seria incomparavelmente maior. O mesmo pode ser aplicado a um projeto de software.

Por meio de conceitos, objetos, símbolos e diagramas, a UML disponibiliza uma forma simples, mas objetiva e funcional, de documentação e entendimento de um sistema. Por exemplo, você pode utilizar os diagramas e arquivos que compõem um modelo UML para o desenvolvimento, apresentação, treinamento e manutenção durante todo o ciclo de vida da sua aplicação. A UML é mais completa que outras metodologias usadas para a modelagem de dados, pois possui, em seu conjunto, os recursos necessários para suprir as necessidades de todas as etapas que compõem um projeto, proporcionando um total controle do projeto.

Apesar de a UML implementar uma modelagem com uma visão orientada a objetos, isso não significa que a ferramenta e a linguagem para a implementação do modelo sejam, também, orientadas a objetos, mas é recomendado que sejam. Este estudo, no entanto, não irá explorar os conceitos de orientação a objetos, e sim a implementação de um modelo UML simples, para início da documentação de um sistema, utilizando dois diagramas implementados pela UML. Entre os diagramas que serão abordados estão o **diagrama de casos de uso** e o **diagrama de classes**.

Dentro desse conceito, cabe discutir o que os diagramas têm como objetivo representar como o sistema irá funcionar e como cada peça do sistema trabalhará e irá interagir com as outras peças do sistema. Existem inúmeras ferramentas para modelagem de dados orientados a objetos, entre as quais, podemos citar o Rational Rose, o Model Maker e o Poseidon UML, o que garante uma vantagem, devido à facilidade de leitura dos diagramas, além da facilidade de confeccioná-los. Além dos diagramas citados, a UML disponibiliza outros diagramas, entre os quais, podemos mencionar o Diagrama de objetos, Diagrama de sequência, Diagrama de colaboração, Diagrama de estado, Diagrama de atividade e Diagrama de componentes.

Reflexão

'UML (Unified Modeling Language) é uma família de notações gráficas, apoiada por um metamodelo único, que ajuda na descrição e no projeto de sistemas de software, particularmente daqueles construídos utilizando o estilo orientado a objetos (OO). Essa definição é um tanto simplificada. Na verdade, para diferentes pessoas a UML tem significados diferentes. Isso ocorre devido à sua própria história e às diferentes maneiras de ver o que compõe um processo de engenharia de software eficaz. Como resultado, em grande parte deste capítulo, minha tarefa é armar o cenário para este livro, explicando as diferentes maneiras pelas quais as pessoas vêm e utilizam a UML.

As linguagens gráficas de modelagem existem há muito tempo na indústria do software. O propulsor fundamental por trás de todas elas é que as linguagens de programação não estão em um nível de abstração suficientemente alto para facilitar as discussões sobre projeto.

Apesar de as linguagens gráficas de modelagem existirem há muito tempo, há uma enorme controvérsia sobre seu papel na indústria de software. Essas controvérsias afetam diretamente o modo como as pessoas percebem o papel da UML em si. A UML é um padrão relativamente aberto, controlado pelo OMG (Object Management Group), um consórcio aberto de empresas. O OMG foi formado para estabelecer padrões que suportassem interoperabilidade, especificamente a de sistemas orientados a objetos. Talvez, o OMG seja mais conhecido pelos padrões CORBA (Common Object Request Broker Architecture).

A UML nasceu da unificação das muitas linguagens gráficas de modelagem orientadas a objetos que floresceram no final dos anos oitenta, início dos noventa. Desde sua aparição, em 1997, ela fez com que essa torre de Babel fosse resolvida. Trata-se de um serviço pelo qual eu e muitos outros desenvolvedores estamos profundamente agradecidos.'

FOWLER (2011, p. 25)

FOWLER, Martin. UML Essencial. [Digite o Local da Editora]: Grupo A, 2011. E-book. ISBN 9788560031382. Disponível em:
<https://integrada.minhabiblioteca.com.br/#/books/9788560031382/>. Acesso em: 19 set. 2022.

Conforme o trecho do texto anterior, observa-se que a UML é resultado de um momento em que havia diferentes formas de modelar software orientado a objetos. Assim, era pouco efetiva a comunicação entre a equipe de desenvolvimento. Vamos conhecer como essa linguagem favorece a produtividade da análise e projeto orientado a objetos?

Ao final deste conteúdo, você será capaz de:

- Reconhecer a notação da linguagem UML.
- Elaborar abstrações de acordo com os componentes da UML.
- Construir modelos UML para visões estáticas e dinâmicas do sistema.



DIAGRAMA de Casos de Uso

O modelo de casos de uso é um dos modelos da UML, composto do diagrama de casos de uso, que apresentam um conjunto de casos de uso, atores e relacionamentos, e de um detalhamento do diagrama de casos de uso, geralmente, uma descrição textual.

O modelo de casos de uso é construído durante as discussões entre os desenvolvedores do sistema de software e os envolvidos (clientes, usuários, áreas da empresa envolvidas direta ou indiretamente no sistema) para especificar os requisitos.

Um modelo de casos de uso apresenta os requisitos funcionais pretendidos do sistema e os elementos externos que interagem com o sistema.

Objetivo principal do modelo de casos de uso

Especificar, visualizar, construir e documentar o comportamento que cada parte do sistema deve ter.

Os casos de uso descrevem os **objetivos** do sistema e não as **funções**!

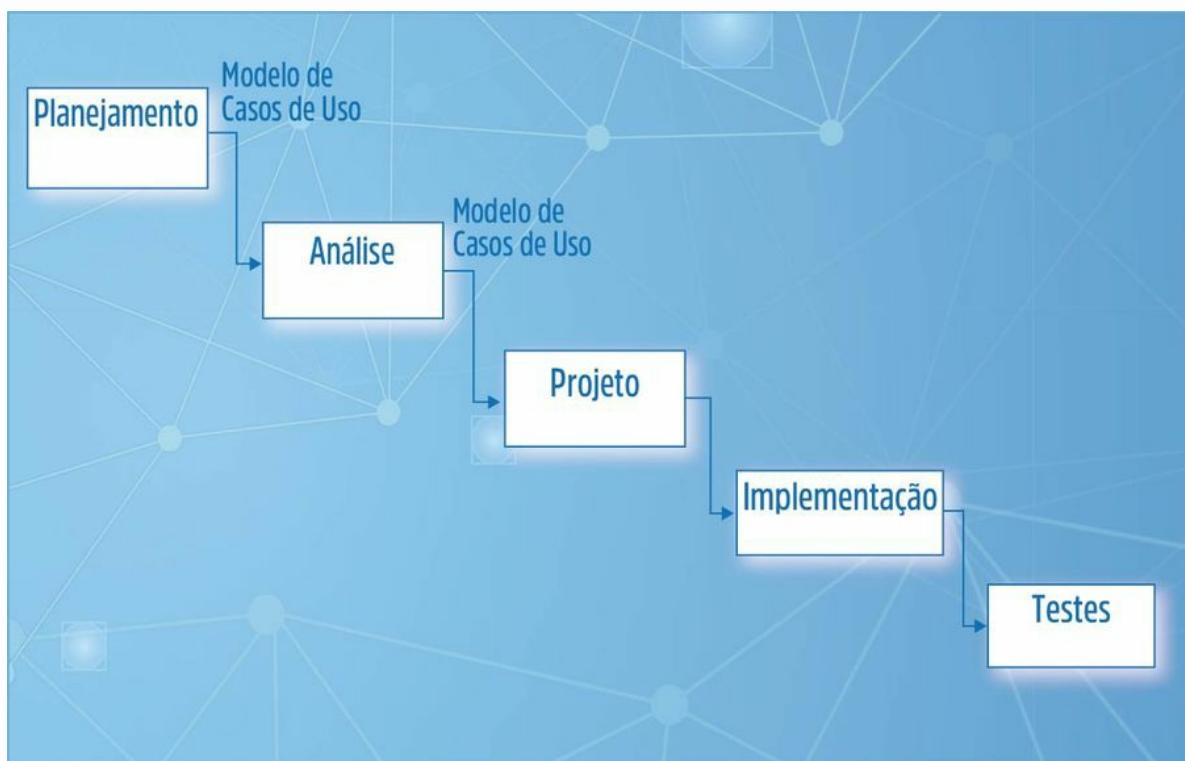
Clique no recurso para visualizar o texto.

Detalhando o objetivo principal, obtêm-se objetivos específicos, que são:

- descrever os requisitos funcionais do sistema, de acordo com todos os envolvidos e os desenvolvedores do sistema de software;
- fornecer uma clara e consistente descrição do escopo do sistema e o que ele deve fazer;
- fornecer requisitos para elaboração do diagrama de classes;
- simplificar alterações e extensões (manutenção) do sistema e verificar o impacto por meio dos casos de uso afetados e novos;
- fornecer cenários para validação dos testes do sistema;
- definir o perfil dos usuários que irão utilizar o sistema;
- definir as condições de início, término e exceções de cada função do sistema; validar outros diagramas da UML.

O diagrama de casos de uso é elaborado nas fases de planejamento e/ou análise, através da ferramenta CASE System Architect, Rational Rose, dentre outras. O diagrama auxilia, também, nas fases de projeto, implementação e testes.

FIGURA 1 - Fases de projeto - Diagrama de caso de uso



SERRA, 2019 [ADAPTADA].

Notação do diagrama de caso de uso

O diagrama de casos de uso é composto, basicamente, por:

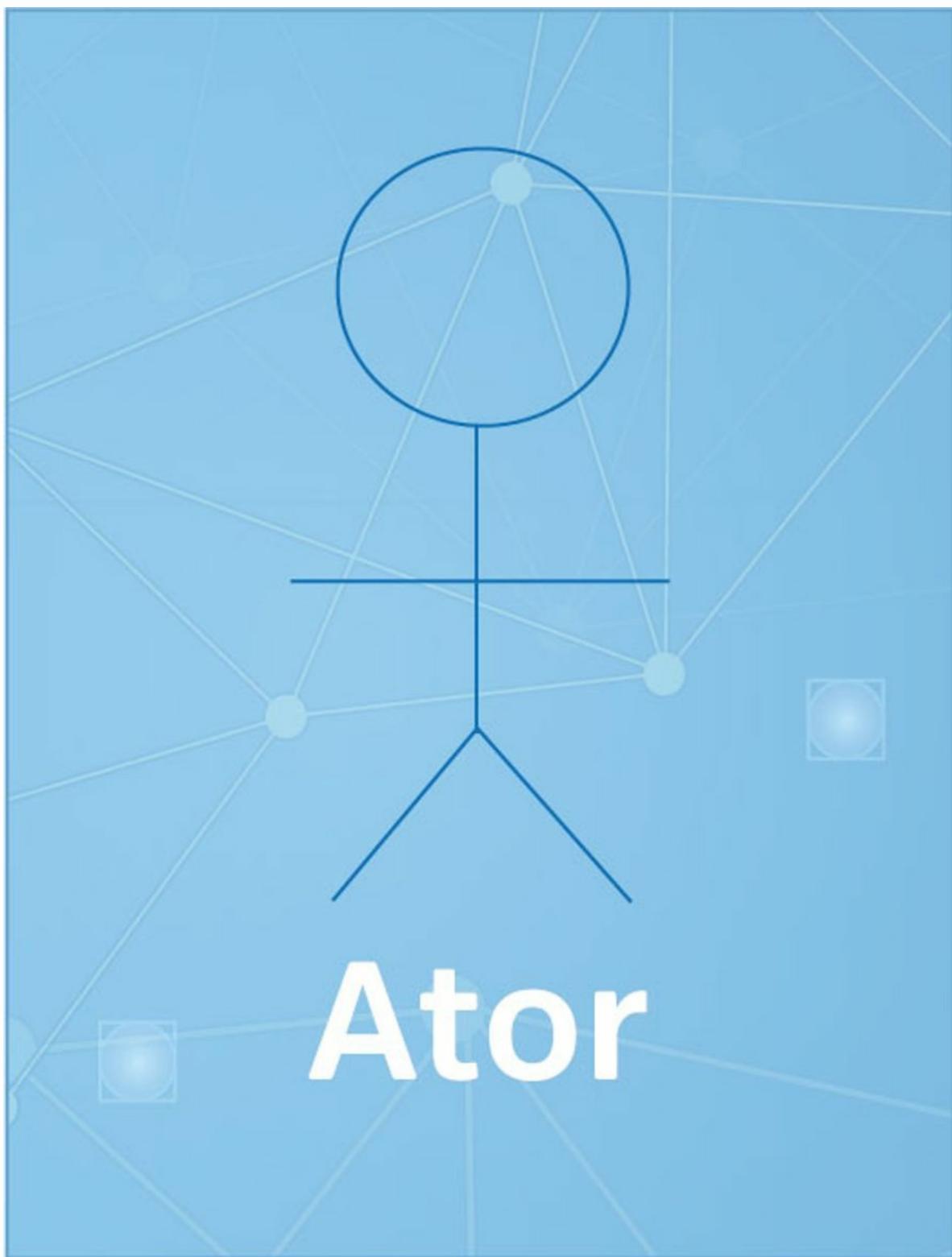
FIGURA 2 - Caso de uso



INSTITUCIONAL, 2022.

Um caso de uso é um conjunto de ações que o sistema realiza para produzir um resultado observável para um ator (objetivos do sistema).

FIGURA 3 - Ator



INSTITUCIONAL, 2022.

São elementos externos ao sistema, que interagem com os casos de uso. Consideram-se atores “alguém” ou “alguma coisa” que não faz parte do sistema, mas que interage com ele.

É o papel executado por alguém e não pela pessoa em si. Por exemplo: José é o administrador do sistema e também usuário comum. Os atores serão o administrador e o usuário, e não a pessoa José.

Exemplos de atores: usuários diretos do sistema, sistemas externos que irão interagir com o sistema ou hardware específico.

Associação é um relacionamento entre o ator e o caso de uso.

FIGURA 4 - Simbologia de associação

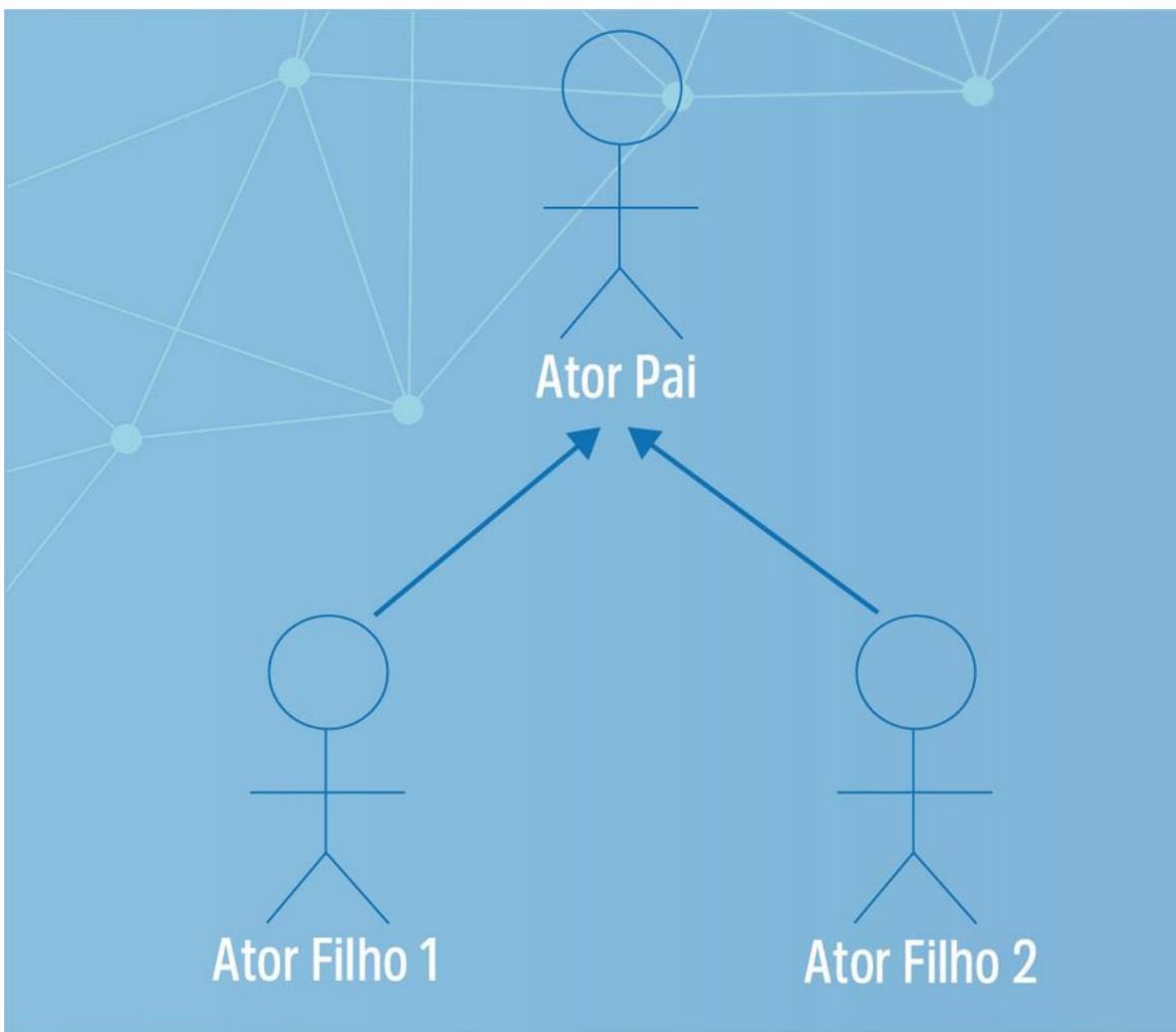


INSTITUCIONAL, 2022.

Notação complementar do diagrama de caso de uso

- Relacionamento entre atores:

FIGURA 5 - Exemplo da notação de generalização



INSTITUCIONAL, 2022.

Atores podem ter um relacionamento de generalização, representando que possuem características e ações comuns, além de algumas características e ações adicionais diferenciadas. A generalização é representada por uma linha contínua e uma seta em formato de triângulo. Essa seta deve ficar apontada para o ator pai (com características e ações comuns) e a outra parte da linha, no ator filho, que herda as características e ações comuns do pai e ainda possui características e ações específicas.

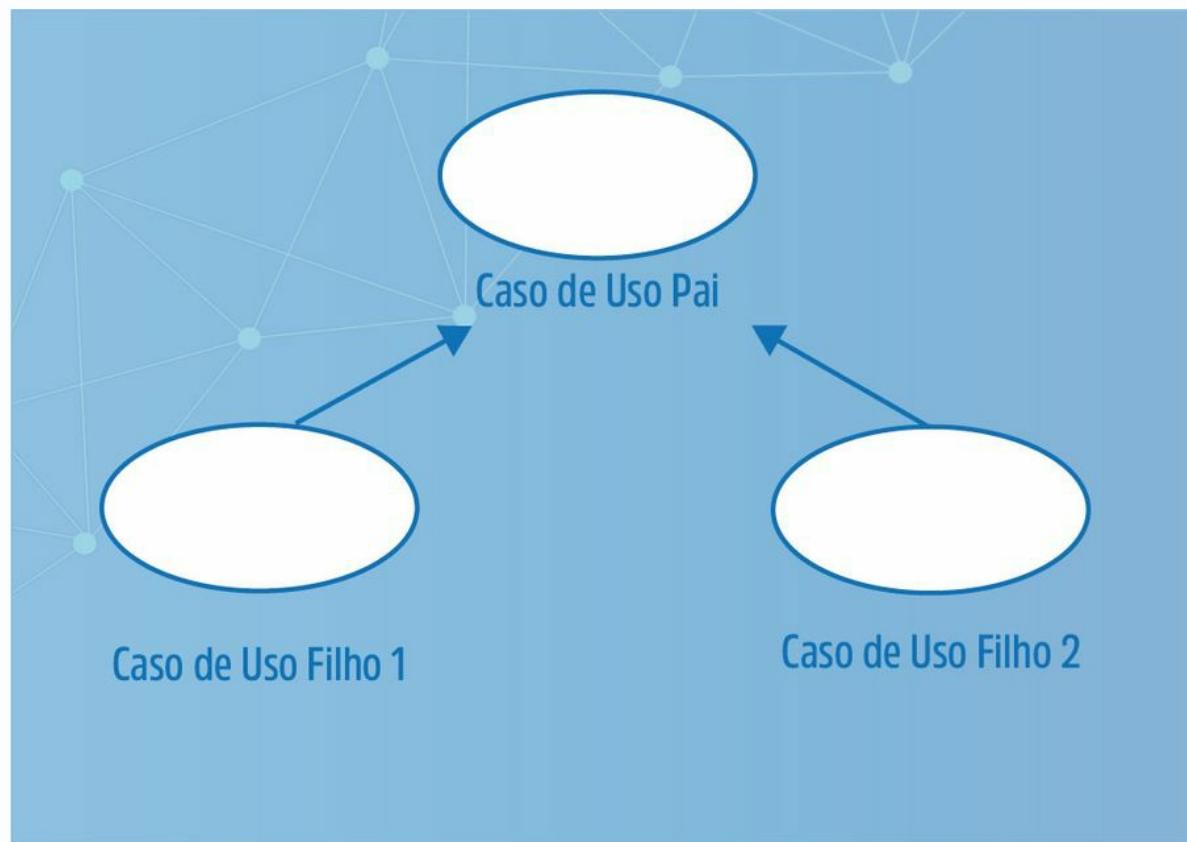
Uma generalização de ator de um tipo de ator (descendente) para outro tipo de ator (ascendente) indica que o descendente herda o papel que o ascendente pode desempenhar em um caso de uso.

- Relacionamento entre casos de uso:

Os casos de uso podem ter relacionamentos entre si. Essas relações devem ser analisadas somente **após a especificação dos casos de uso**, pois, com a descrição detalhada dos casos de uso, podemos verificar possíveis relacionamentos entre eles.

Basicamente, existem três tipos de relacionamento entre casos de uso, que são:

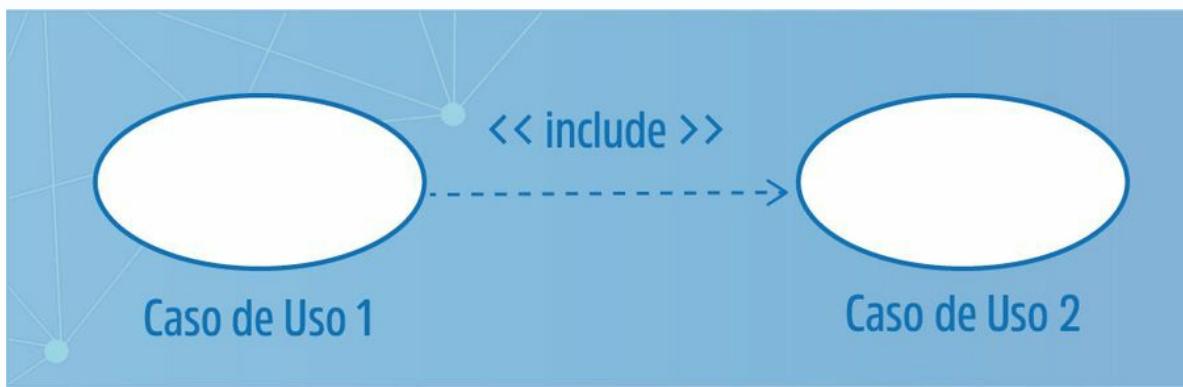
FIGURA 6 - Generalização



INSTITUCIONAL, 2022.

Como no relacionamento de generalização entre atores, o relacionamento de generalização entre casos de uso é utilizado quando existem dois ou mais casos de uso que têm comportamento, estrutura e finalidade comuns. Quando isso ocorre, pode descrever as partes compartilhadas em um caso de uso novo, que é especializado pelos casos de uso filho.

FIGURA 7 - Inclusão (Include)

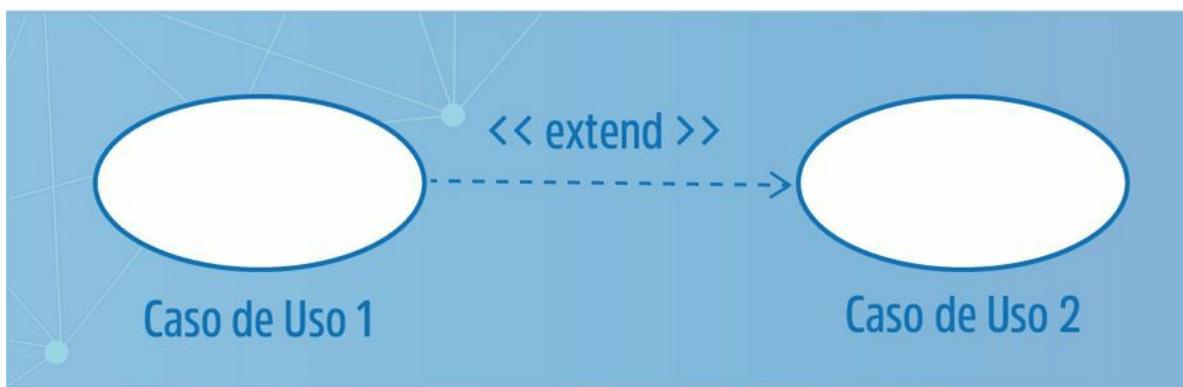


INSTITUCIONAL, 2022.

Relacionamento que indica que um caso de uso contém comportamento definido em outro caso de uso, e que toda vez que o caso de uso base for executado, o caso de uso de inclusão relacionado pelo <<include>> também será executado. O relacionamento de inclusão pode ser usado para:

- Separar o comportamento do caso de uso base que não seja necessário para compreender a finalidade principal do caso de uso; apenas o resultado é importante;
- Separar o comportamento que seja comum a dois ou mais casos de uso.

FIGURA 8 - Extensão (Extend)



INSTITUCIONAL, 2022.

Relacionamento que especifica que o comportamento de um caso de uso base pode ser estendido para um outro caso de uso adicional. A extensão é condicional, o que significa que sua execução depende do que tiver acontecido durante a execução do caso de uso base. O relacionamento de extensão pode ser utilizado para:

Mostrar que uma parte de um caso de uso é um comportamento opcional (ou possivelmente opcional) do sistema. Isso faz a diferenciação entre comportamento opcional e comportamento obrigatório em um modelo;

Mostrar que um subfluxo só é executado em determinadas condições excepcionais;

Mostrar que pode haver um conjunto de segmentos de comportamento entre os quais um ou vários podem ser inseridos em um ponto de extensão de um caso de uso base. Os segmentos de comportamento que são inseridos (e a ordem na qual são inseridos) dependerão da interação com os atores durante a execução do caso de uso base;

Podem representar os fluxos alternativos relevantes de um determinado caso de uso. Princípio da extensão:

Ele sempre adiciona comportamento a um caso de uso;
O caso de uso base deve continuar intacto e tendo valor por si próprio;
O comportamento básico do caso de uso estendido deve sempre ficar intacto.

Passo a passo para a elaboração do diagrama de caso de uso

A seguir, são apresentados os passos que podem ser realizados para a elaboração do diagrama de caso de uso.

Tutorial

00

Passo 1

Identificação dos atores

Listar os possíveis usuários diretos do sistema, os sistemas externos e hardwares específicos. Para isso, pode-se identificar os possíveis atores e objetivos em relação ao sistema (atores x objetivo).

Dicas para encontrar atores:

- Inicie identificando os atores;
- Trabalhe do específico para o geral;
- Não esqueça os atores de suporte;
- Considere as informações que você possui (documento de visão, por exemplo);
- Lembre que atores nem sempre são pessoas;
- Foque na fronteira do sistema;
- Identifique as fontes de informação;
- Evolua os atores com os casos de uso.

Perguntas para ajudar:

- Quem vai fornecer, usar ou remover informações?
- Quem usará essa funcionalidade?
- Quem está interessado em um determinado requisito?
- Em que parte da organização o sistema é usado?
- Quem vai dar suporte e manter o sistema?
- Quais são os recursos externos do sistema?
- Que outros sistemas precisarão interagir com este?

01

Passo 2

Identificação dos casos de uso

Listar os possíveis **objetivos** do sistema. Para identificar esses objetivos, algumas questões podem ser feitas:

- De quais objetivos do sistema o ator necessita? O que o ator precisa fazer?
- O que o ator precisa fazer? O ator precisa calcular, consultar, criar, destruir, modificar ou registrar algum tipo de informação? Por quê?
- O ator deve ser notificado sobre eventos do sistema ou precisa notificar o sistema sobre algo? O que esses eventos representam em termos de funcionalidade?
- Existem objetivos de comunicação com outros sistemas?

- O ator precisa estar informado sobre certas ocorrências no sistema?
- Que informações devem ser modificadas ou criadas no sistema?

Complementando:

- Não se preocupe com partes comuns (pelo menos, no início);
- Não confundir casos de uso com “funções”;
- Foque SEMPRE no valor para o ator;
- Não esqueça os casos de uso de suporte e operacionais;
- Evolua os casos de uso junto com os atores.

02

Passo 3

Associação entre os atores e casos de uso

Elaboração propriamente dita do diagrama de caso de uso. Deve-se verificar se todo ator possui, pelo menos, uma associação com um caso de uso e se todo caso de uso interage com algum ator.

Descrição do diagrama de casos de uso

Além da notação gráfica, o diagrama de casos de uso deve possuir uma especificação de detalhamento do diagrama de caso de uso. O modelo padrão do RUP (Rational Unified Process) é que será utilizado nas aulas, e aborda os seguintes itens.

- catálogo de atores, contendo o nome do ator e uma descrição de cada ator representado no diagrama;
- nome do caso de uso;
- breve descrição do caso de uso;
- fluxo básico (detalhar o passo a passo do caso de uso);
- fluxos alternativos (qualquer exceção que ocorra nos passos do fluxo básico deve ser detalhado como fluxo alternativo);
- pré-condições (lista de condições que devem ser verificadas antes que o caso de uso comece);
- pós-condições (lista de condições que devem ser verificadas depois do fim do caso de uso).

Dicas para escrever casos de uso:

- escreva na voz ativa: “O sistema valida a quantidade entrada” em vez de “A quantidade entrada deve ser validada pelo sistema”;
- escreva no tempo presente;
- descreva as interações que ocorrem entre os atores e o sistema;
- Use subfluxos para simplificar descrições complexas;
- não encha seu modelo de casos de uso com o CRUD (Create, Read, Update, Delete);
- não tenha medo de capturar os detalhes.

Saiba Mais

Caso queira, poderá acessar a um tutorial de caso de uso de UML. Indicamos este:

https://www.youtube.com/watch?time_continue=32&v=ab6eDdwS3rA&feature=emb_logo

Considerações e observações

Um caso de uso sempre deve ter um relacionamento com um ator. Isso implica que cada caso de uso deve ter associações de comunicações com os atores. O motivo dessa regra é forçar o sistema a fornecer apenas a funcionalidade de que os usuários precisam e nada mais. Ter casos de uso que ninguém solicita é uma indicação de que algo está errado no modelo de casos de uso ou nos requisitos. Entretanto, há algumas exceções a essa regra:

Um caso de uso filho em um relacionamento de generalização não precisa ter um ator associado a ele se o caso de uso pai está associado a um ator;

Um caso de uso pode ser iniciado de acordo com uma programação (por exemplo, uma vez por semana ou uma vez por dia), o que significa que o relógio do sistema é o iniciador. O relógio do sistema é interno ao sistema, e o caso de uso não é iniciado por um ator, mas por um evento do sistema interno. Se não ocorrer outra interação do ator no caso de uso, ele não terá nenhuma associação com os atores. Contudo, para esclarecer, você pode usar um ator fictício 'tempo' para mostrar como o caso de uso é iniciado nos diagramas de casos de uso.

Na grande maioria das modelagens de diagramas de casos de uso, a notação básica (ator, relacionamento e caso de uso) é suficiente para especificar os requisitos. Isso significa que não necessariamente é obrigatório utilizar herança <<include>> e <<extends>> em toda modelagem de sistemas; inclusive, a notação de <<include>> e <<extends>> pode ser omitida no diagrama e detalhada no passo a passo do fluxo básico e/ou alternativo;

Geralmente, utiliza-se o termo “manter” para indicar ações de incluir, consultar, alterar e excluir. Para isso, pode-se utilizar o estereótipo <<CRUD>> (create, read, update e delete). Com isso, deixamos o detalhamento das funções para a especificação do caso de uso e mantemos o diagrama mais claro.

Após a elaboração do diagrama de caso de uso e das especificações dos casos de uso, deve-se verificar se há a necessidade de criação de generalização entre atores e casos de uso, inclusões (include) e extensões (extend).

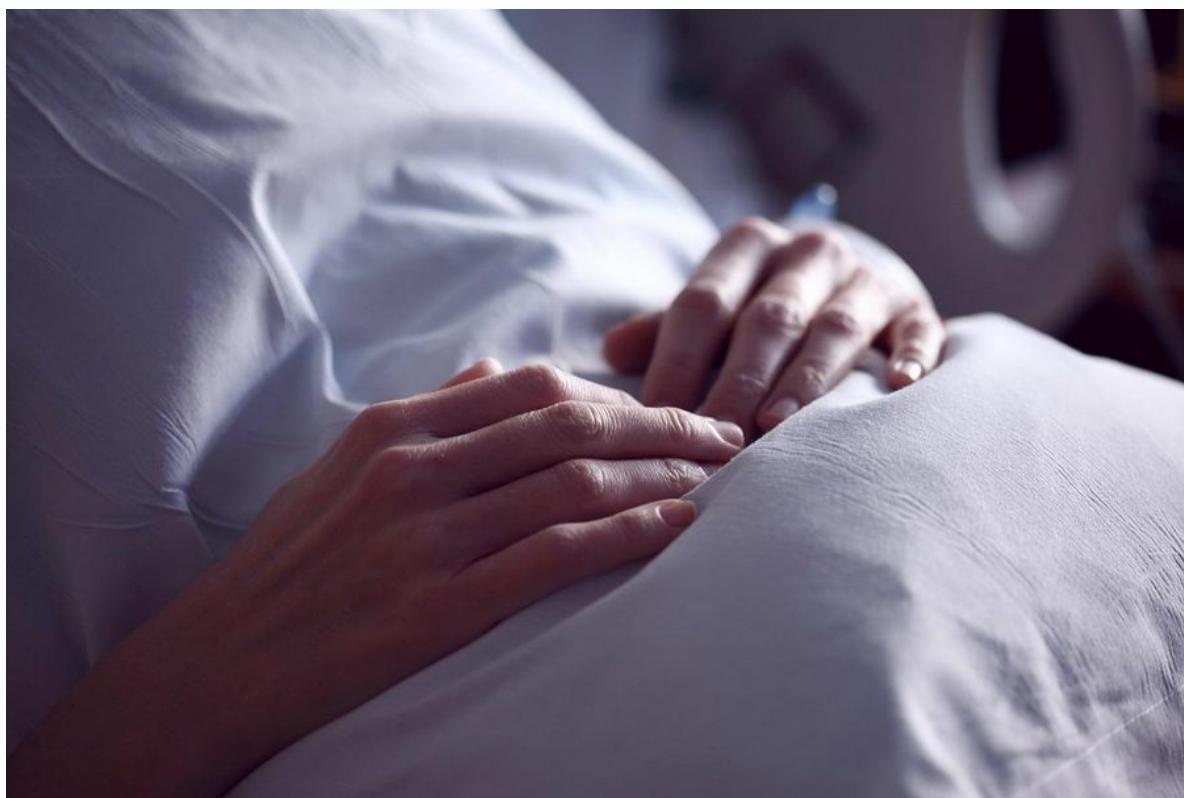
Confira o estudo de caso a seguir.

Estudo de Caso

“A clínica médica Saúde Perfeita precisa de um sistema de agendamento de consultas e exames. Um paciente entra em contato com a clínica para marcar consultas visando realizar um check-up anual com seu médico de preferência. A recepcionista procura a data e hora disponível mais próxima na agenda do médico e marca as consultas. Posteriormente, o paciente realiza a consulta, e, nela, o médico pode prescrever medicações e exames, caso necessário.”

Com esse cenário simples, começaremos a criar o diagrama de caso de uso. Inicialmente, iremos definir os atores, que serão:

FIGURA 9 - Paciente



BANCO DE IMAGEM

FIGURA 10 - Secretária



BANCO DE IMAGEM

FIGURA 11 - Médico



BANCO DE IMAGEM

Agora, definiremos as ações de cada usuário:

Paciente

- Solicita consulta;
- Solicita cancelamento de consulta.

Secretária

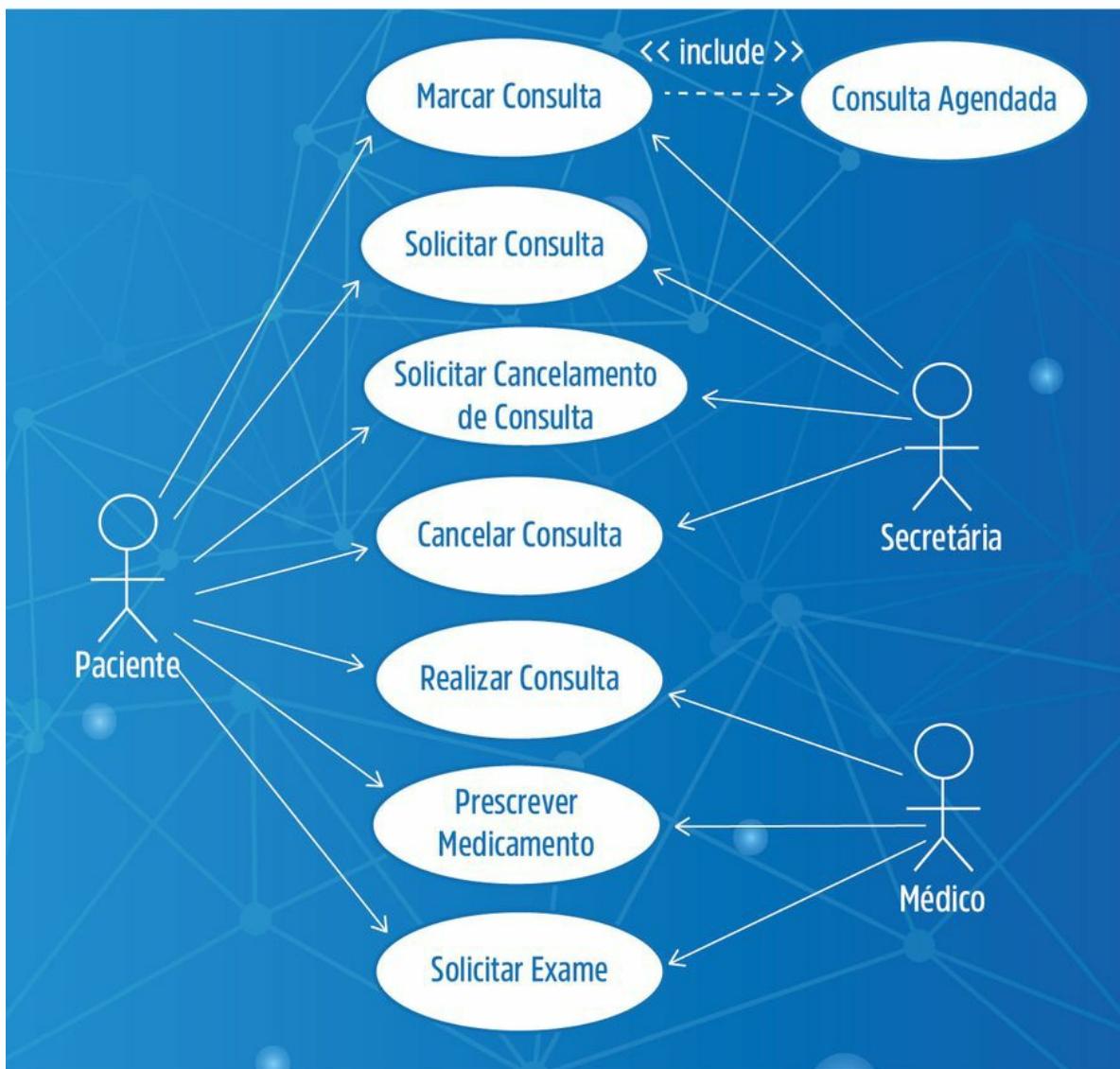
- Consulta agenda;
- Marca consulta;
- Cancela consulta;

Médico

- Realiza consulta;
- Prescreve medicação;
- Solicita realização de exames.

Depois que definirmos os atores e suas ações, podemos criar o diagrama de caso de uso que irá nos demonstrar, de uma forma clara, a interação entre atores e ações.

FIGURA 12 - Diagrama de caso de uso - Estudo de caso 'Clínica'



SERRA, 2019 [ADAPTADA].

Diagrama de classes

Descrição

O diagrama de classes é o principal diagrama da UML, e sendo estático, modela a estrutura do sistema, por meio de atributos, operações e relacionamentos. Uma classe é a descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica.

Aplicação

O diagrama de classes é utilizado para:

- Propor a estrutura do sistema;
- Auxiliar na definição de componentes e na implementação.

Palavras-chave: estrutura do sistema.

O diagrama de classes é um dos produtos das fases de análise e/ou projeto e pode ser elaborado através de ferramentas CASE. Esse diagrama é o fundamento da modelagem orientada a objetos, auxiliando em toda modelagem, definição dos componentes e codificação.

O diagrama de classes pode ser classificado em dois tipos:

Diagrama de classes de análise

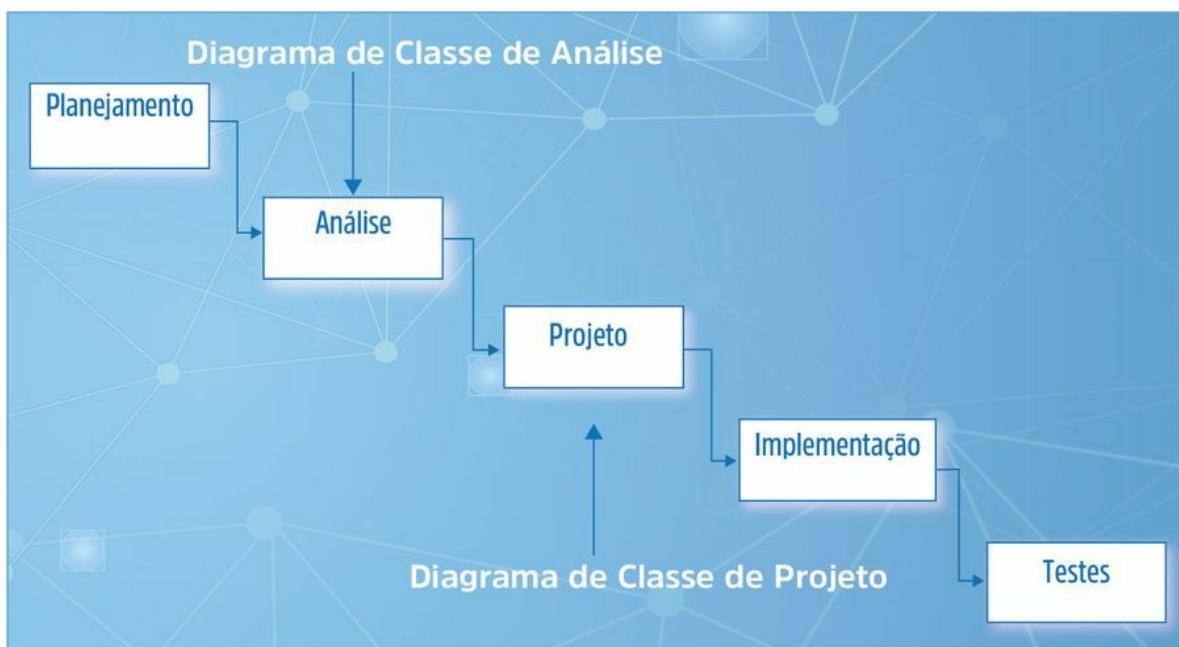
Representam as classes de negócio do sistema, sendo um primeiro modelo conceitual para representar elementos do sistema que possuam características, ações e responsabilidades. Com o tempo, as classes de análise evoluem e se transformam em classes de projeto. As classes de análise não devem se preocupar diretamente com a implementação, ou seja, com particularidades da linguagem de programação;

Diagrama de classes de projeto

Representam as classes de negócio e de implementação do sistema. Para isso, o diagrama de classes de análise deve ser mapeado para um diagrama de projeto que deve abranger as classes de análise, os componentes de implementação, as interfaces externas, as particularidades relevantes da linguagem de programação e da arquitetura do sistema. Esse diagrama é elaborado com base na “realização dos casos de uso”.

Confira a figura a seguir, que exemplifica.

FIGURA 13 - Fases de projeto - Diagrama de classes



SERRA, 2019 [ADAPTADA].



ATENÇÃO

Alguns autores, como Scott e Fowler, classificam os diagramas de classes em 3 tipos, sendo:

- 1) Conceitual, para definir as classes de negócio;
- 2) Especificação, para definir as interfaces; e
- 3) Implementação, para definir as classes de implementação.

Para nossas aulas, o diagrama conceitual equivale ao diagrama de classes de análise, e os diagramas de especificação e de implementação serão desenvolvidos em um único, que será o diagrama de classes de projeto.

Notação básico- Diagrama de classes de análise

Classe x Objeto

Classe:

- Representa algo do mundo real;
- Define as características e o comportamento dos objetos.

Objeto:

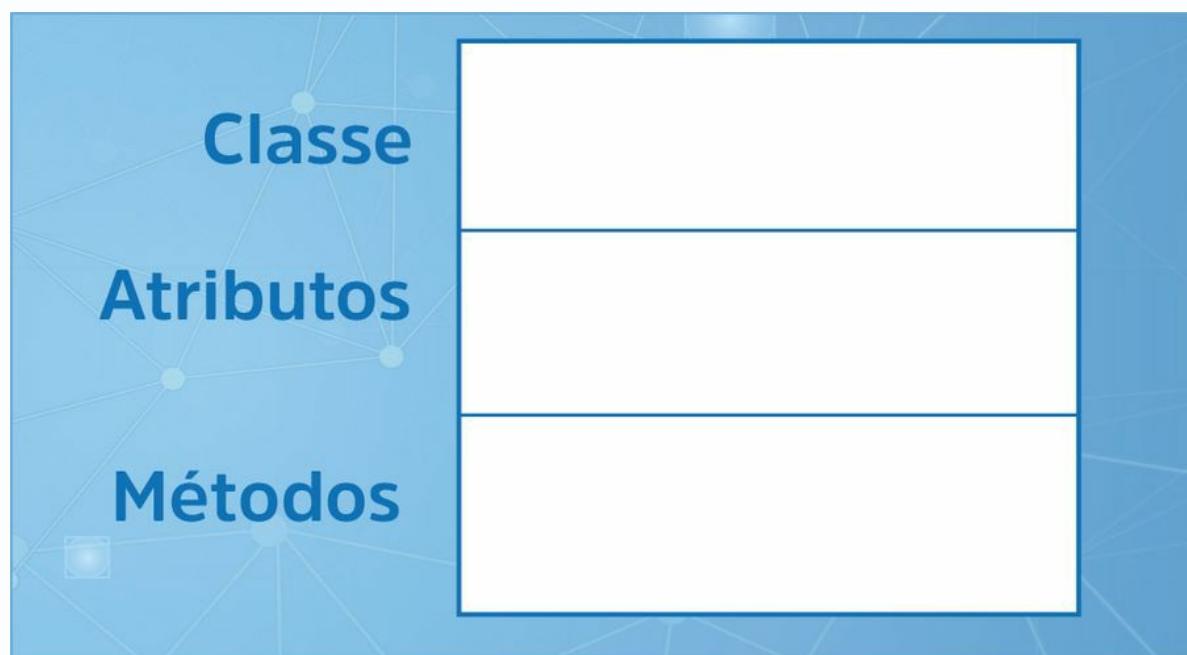
- Uma ocorrência de uma classe;
- Criado durante a execução do sistema.

Classe

- Classe é um conjunto de objetos que possuem características e comportamentos comuns;
- Podem ser qualquer elemento que seja relevante dentro do domínio do problema;
- Iniciar com letra maiúscula e no singular;
- Podem ter nomes compostos.

Uma classe é representada por um retângulo composto de nome da classe, atributos e métodos (ações).

FIGURA 14 - Exemplo de uma classe



SERRA, 2019 [ADAPTADA].

Nome da classe

O nome da classe deve expressar o que a classe representa, podendo ser um texto (substantivo) simples ou composto no singular e iniciado com letra maiúscula.

Atributos

Um atributo é uma propriedade (características) da classe. Em determinado momento, um objeto de uma classe terá um valor específico para cada um dos atributos de sua classe.

- Descrevem as características dos objetos;
- Definem os dados que devem ser armazenados;
- São as informações que uma classe deve ter de si mesma;
- Exemplo de classe 'Cliente': nome, cnpj, inscrição estadual.

Métodos

Uma operação é uma ação ou transformação realizada ou sofrida por um objeto. Muitas vezes, utiliza-se o termo 'método' em vez de 'operação'. Método é a implementação de uma operação para uma classe.

- São utilizados para:
 - Manipular os atributos;
 - Realizar outros tipos de ações.
- Pertencem às classes e somente podem ser aplicados aos objetos da classe;
- Definem os serviços que a classe pode oferecer.



ATENÇÃO

Os níveis de acessibilidade de um atributo e/ou um método podem ser:

- Pública (+): objetos de quaisquer classes podem acessar o atributo ou o método;
- Privada (-): apenas a classe possuidora do método ou do atributo pode ter acesso;
- Protegida (#): apenas a classe e as subclasses possuidoras do método ou do atributo podem ter acesso.

Relacionamentos

Um relacionamento é uma conexão entre classes que representa comunicação e interação entre classes.

Os tipos de relacionamentos que podem ser usados são associação (simples, agregação e composição), generalização e dependência.

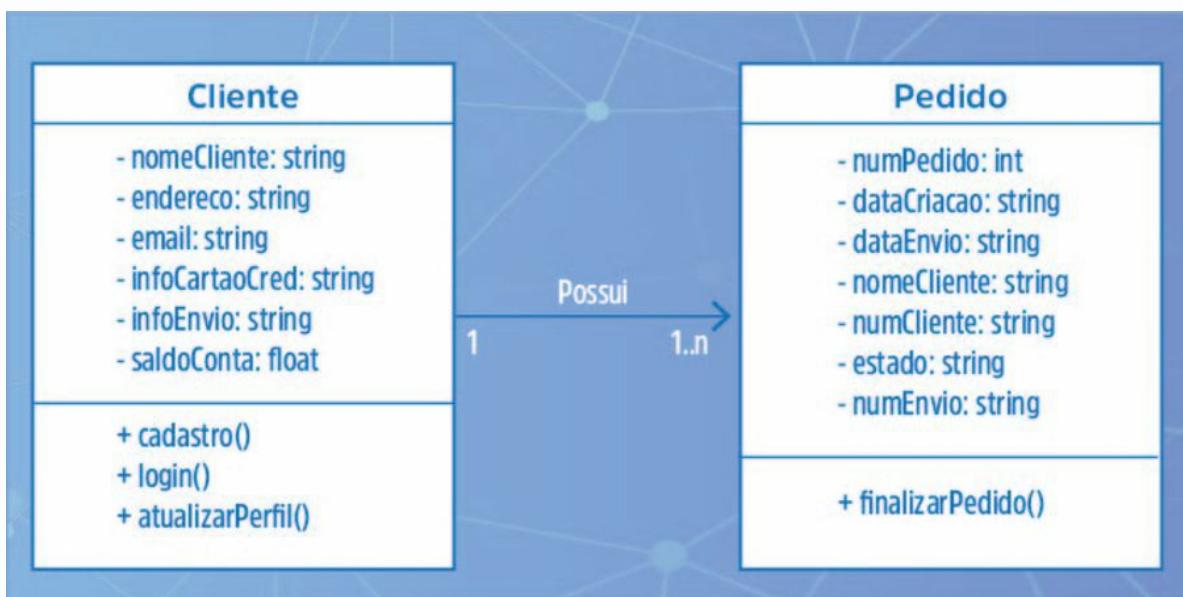
Associação: é um relacionamento que demonstra que uma classe está conectada a outra, com o intuito de comunicação entre elas. A associação, normalmente, é bidirecional. Isso significa que, se um objeto é associado a outro objeto, ambos os objetos se “conhecem”. A associação simples é uma conexão entre classes. Isso é representado por uma linha contínua entre as duas classes. A associação deve possuir um nome, representado próximo à linha de associação, e frequentemente, é um verbo. Também é sugerido que a associação possua uma seta, que indica a direção de leitura. Tanto o nome da associação quanto a seta são importantes para facilitar o entendimento do domínio do problema.

Na associação, deve-se especificar a cardinalidade entre as classes. A cardinalidade é a indicação de quantos objetos podem participar de um relacionamento, ou seja, a multiplicidade. A multiplicidade das associações define todas as possibilidades numéricas de ocorrências entre os objetos.

undefined

A associação e a cardinalidade entre as classes estão diretamente relacionadas ao domínio do problema. A seguir, o diagrama de classe representa que uma cliente possui um ou n pedidos e que um pedido pertence a somente um cliente.

FIGURA 15 - Exemplo de diagrama de classe



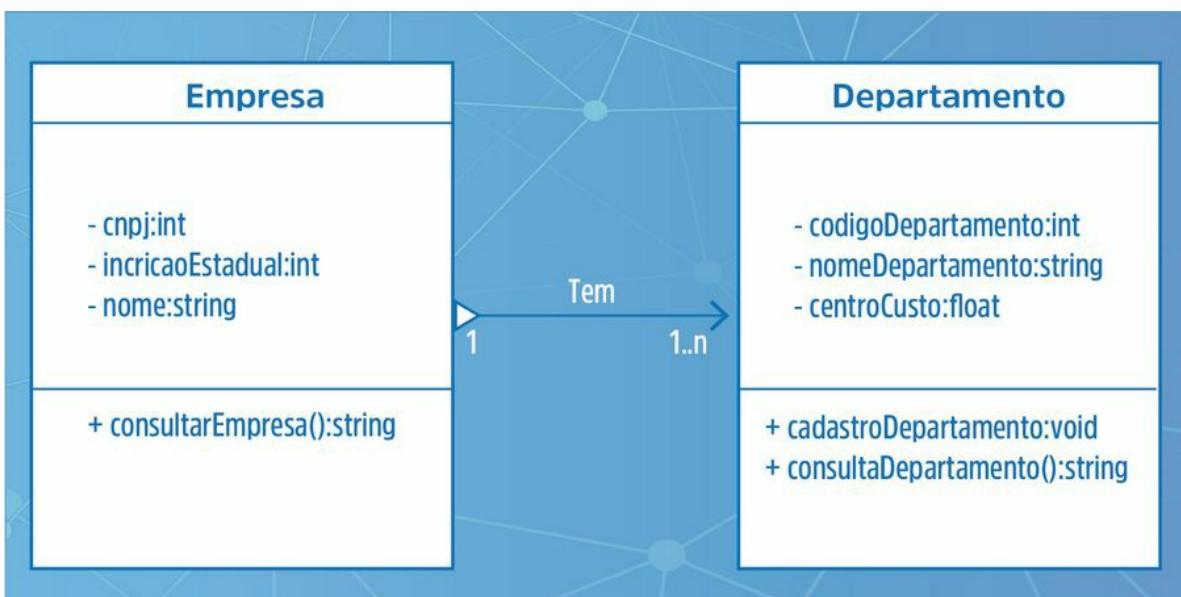
SERRA, 2019 [ADAPTADA].

Além da associação simples, apresentada acima, existem dois tipos específicos de associação:

Agregação: é um caso especial de associação que representa, conceitualmente, que uma ou mais classes fazem parte de uma outra classe (“todo parte”). Contudo, as classes “todo parte” podem ‘viver’ independentemente. A notação da agregação é um losângulo sem preenchimento.

O diagrama de classes abaixo representa que uma empresa pode ser composta de um ou mais departamentos, sendo que, não necessariamente, ao instanciar a classe Empresa, a classe Departamento deve ser instanciada.

FIGURA 16 - Exemplo de diagrama de classes - Agregação

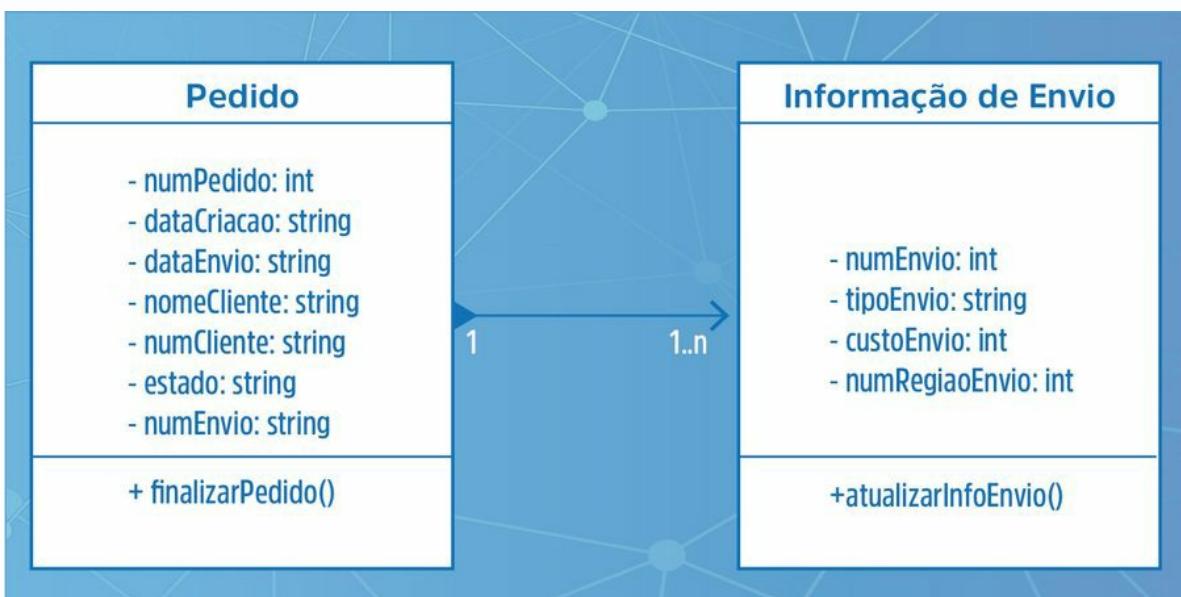


SERRA, 2019 [ADAPTADA].

Composição: é um caso especial de associação e uma forma de agregação mais forte, em que o objeto parte depende do objeto todo, ou seja, espera-se que as partes “vivam” e “morram” na totalidade, conjuntamente. A notação da agregação é um losângulo preenchido.

O diagrama de classes a seguir representa que a classe Pedido é composta de uma ou mais classe Informação de envio, aendo que ao instanciar a classe Pedido, a classe Informação de Envio será instanciada “automaticamente”.

FIGURA 17 - Exemplo de diagrama de classe - Composição



INSTITUCIONAL, 2022.



ATENÇÃO

Tanto no caso da agregação como no da composição, o losângulo fica sempre na classe principal. Todas as regras de associação (por exemplo, nome da associação, seta da associação e cardinalidade) são válidas para agregação e composição.

Generalização: é um relacionamento que descreve uma relação entre grupos de “coisas” com algo em comum, e demonstra a herança de atributos e operações de uma classe pai para seus filhos. A generalização é utilizada em classes visando à reutilização de implementação.



ATENÇÃO

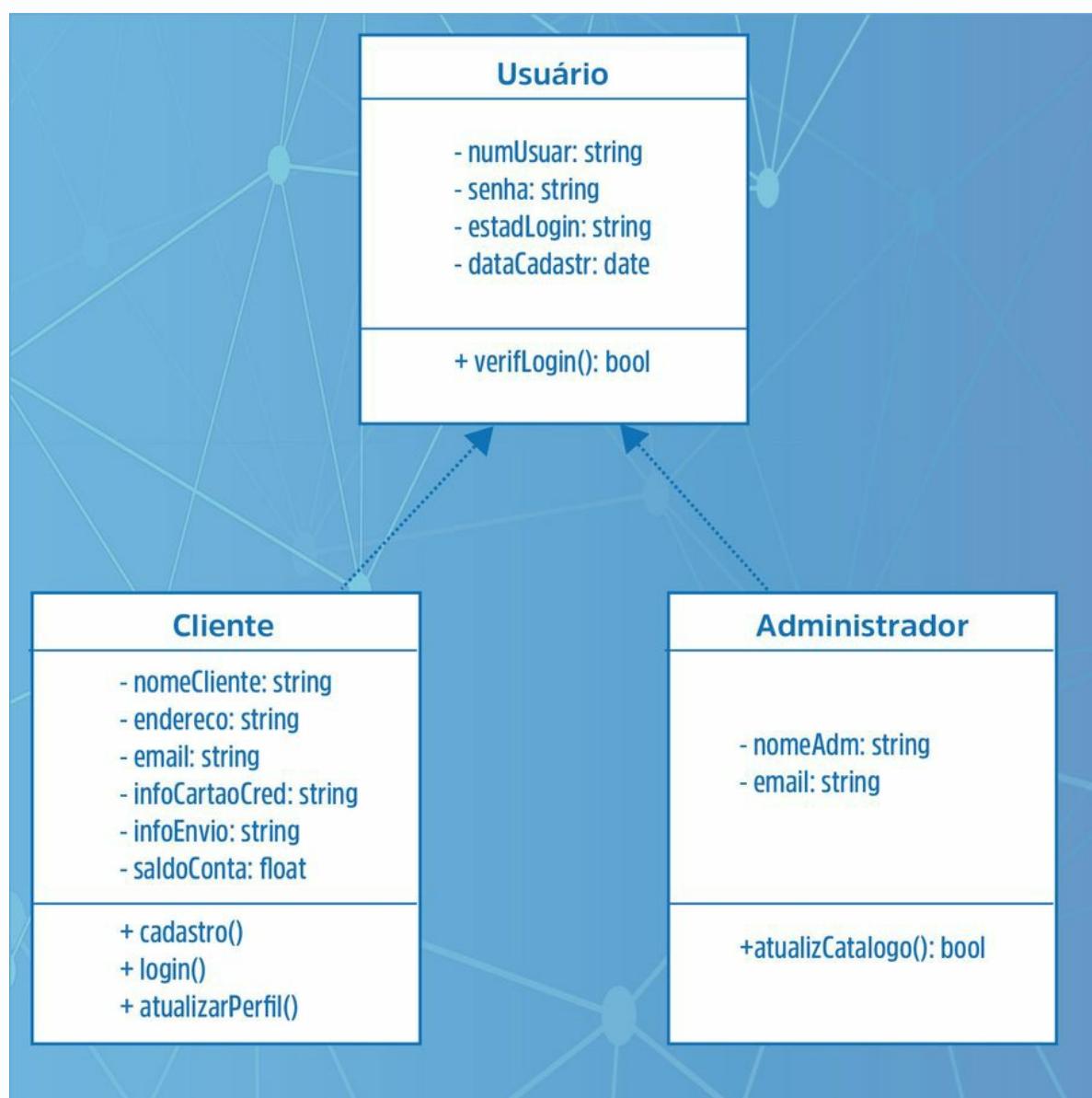
O “triângulo” (representação da generalização) deve ser representado na classe pai. Não existe cardinalidade no relacionamento de generalização.

me

operações que sejam comuns às classes originais. Estes, por sua vez, são reorganizados para conter somente a parte especializada. A especialização é um processo top-down e é utilizada para refinar o modelo. A partir de classes existentes no modelo, procuram-se especializações para o mesmo, como casos particulares da classe genérica, definindo atributos e operações específicos.

O diagrama de classes a seguir representa que um usuário pode ser um cliente ou um administrador. Tanto o cliente como os administradores possuem características (atributos) e ações (operações) comuns, que são representados na classe pai Usuário. E as características (atributos) e ações (operações) específicas ficam nas classes filhas (Cliente e Administrador).

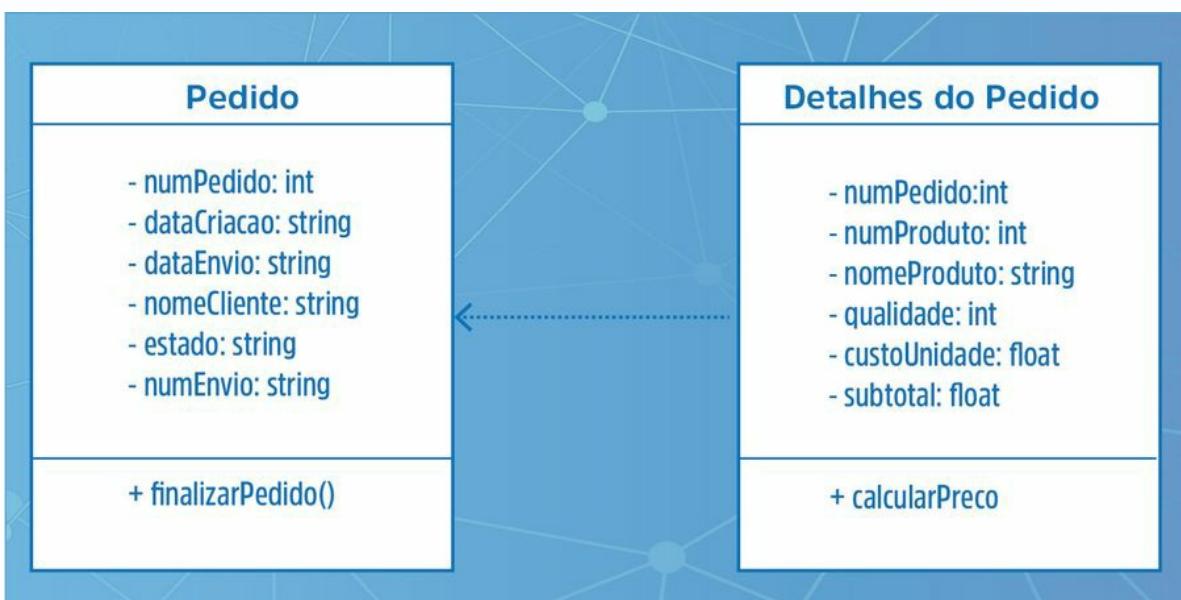
FIGURA 18 - Exemplo de diagrama de classes - Generalização



Dependência: é um relacionamento de utilização, determinando que uma classe utilize atributos e operações de outra classe, mas não necessariamente, o inverso.

A classe "Detalhes do pedido" depende da classe "Pedido", pois se considera, nesse exemplo, que, ao listar um pedido, os detalhes do pedido devem ser exibidos.

FIGURA 19 - Exemplo de diagrama de classes - Dependência



INSTITUCIONAL, 2022.

Procedimento para a elaboração do diagrama de classes de análise

Identificação das classes

Descrição

A primeira fase da elaboração de um modelo de classes consiste na identificação das classes que compõem o sistema. A partir da descrição textual do sistema (especificação de requisitos) e/ou dos diagramas e descrições dos casos de uso, serão identificadas as classes relevantes através de sucessivos refinamentos e detalhamentos, ou seja:

- Foco nas classes de negócio;

- Classes se originam de:

- Especificações de requisitos;
- Diagramas de casos de uso;
- Modelos de processo;
- Entrevistas;
- Reuniões.

Confira a seguir, a **estrutura da solução**.

Tutorial

00

1

Identificação

Identifique os substantivos na descrição do sistema (requisitos, especificação de casos de uso, diagrama de casos de uso, etc.);

01

2

Classificação

Anote-os separadamente e classifique-os;

02

3

Avaliação

Refine os objetos identificados, avaliando os que devem permanecer no modelo;

03

4

Esboço

Crie um esboço do diagrama com os objetos identificados;

Não se preocupe com atributos, operações e relacionamentos;

Tente refiná-los, agrupando os objetos fortemente relacionados.



ATENÇÃO

- Perguntas para auxiliar na identificação de classes:
 - Existem informações que devem ser armazenadas?
 - Existem sistemas externos?
 - Existem representações organizacionais?
 - Os atores representam algo no negócio?
 - Existem dispositivos utilizados pelo sistema?
 - Existem bibliotecas, componentes ou outros elementos de software?

- Dirija a modelagem segundo o problema a resolver;
- Escolha os nomes das classes com critério;
- Analise cuidadosamente o texto, examinando, também, pronomes,

- para não omitir substantivos implícitos;
- Não elimine classes a não ser que tenha absoluta certeza. É sempre possível eliminar uma classe mais tarde;
- Planeje a localização de suas classes no diagrama, facilitando enxergar os relacionamentos.

Identificação de associação entre classes

Descrição

Após a identificação das classes que fazem parte do sistema, é necessário relacionar as classes de acordo com os seus papéis. Uma classe isolada no sistema, geralmente, não é útil; portanto, toda classe tem algum tipo de relacionamento com outra classe. É necessário identificar, inicialmente, as associações e agregações entre as classes do sistema e, posteriormente, as generalizações/herança.

Estrutura da solução:

Associações

- Marque os verbos e expressões que caracterizam os substantivos identificados como objetos;
- Identifique as associações entre os objetos;
- Dê nomes às associações;
- Identifique as associações entre as classes. Num primeiro momento, não tente distinguir as agregações entre as associações;
- Num segundo momento, distingua quais associações podem ser agregações ou composições.

Cardinalidades

- Estime a cardinalidade das associações.

Hierarquia de generalização/herança

- No diagrama de caso de uso, os atores que possuem generalização/herança tendem a ser classes representadas através da generalização/herança. Procure, no texto de descrição do sistema e/ou na descrição dos casos de uso, casos de classes especializadas. Algumas classes podem não estar muito explícitas ou podem não parecer que são especializadas. Aproveite para examinar seu

modelo novamente contra a descrição textual, olhando o sistema do ponto de vista desses agrupamentos;

- Tente definir classes generalizadas e agrupar seus casos particulares;
- Refine o modelo, observando as dicas abaixo. Não exagere. Ao descobrir as possibilidades do mecanismo de herança, o entusiasmo é comum e tenta-se ver a generalização em todo lugar. Cuidado para não complicar demais o modelo.



ATENÇÃO

As cardinalidades dizem respeito a cada extremidade de uma associação;

As cardinalidades podem mudar dependendo do contexto em que a associação ocorre, do domínio do problema e das regras de negócio.

Identificação de atributos e operações

A definição dos atributos e operações na classe é baseada nas responsabilidades.

Responsabilidade é um contrato de uma determinada classe.

undefined

Ao realizar a modelagem, um bom ponto de partida para identificar as responsabilidades: técnicas como CRC (Class Responsibility Collaboration) e análises baseadas em casos de uso são de grande ajuda.

Estrutura da solução para atributos

- A partir da descrição textual do problema, tente identificar substantivos e adjetivos que representem propriedades das classes identificadas;
- Tente descobrir mais alguns atributos através de análise e discussão. Atributos podem ser omitidos da descrição, mas o conhecimento do domínio da aplicação traz à tona muitos atributos implícitos;
- Complete o modelo de classes com os atributos encontrados e verifique se estão na classe correta.

Estrutura da solução para operações

- Identifique as possíveis operações que irão manipular os atributos identificados e qual é a classe que será responsável pela execução dessa operação;
- Incluir essas operações nas classes.



ATENÇÃO

Atributos podem ser identificados como novas classes, dependendo do contexto. Se um atributo tem atributos, então, ele é uma classe. Se a existência do atributo parece mais importante que o valor que ele guarda, então, ele é uma classe.

Observe o nível adequado de detalhamento. Alguns atributos são mais importantes que outros. Atributos que afetam muitas operações da aplicação são altamente relevantes. Outros atributos que têm efeito pequeno não necessitam ser considerados na análise.

Cuidado com atributos necessários à implementação. Alguns atributos podem ser necessários somente para a linguagem em que será implementado o aplicativo, mas podem não ter função lógica no domínio da aplicação. Não considere esses atributos no diagrama de classes de análise.

Verifique atributos fora de contexto. Se forem identificados grupos de atributos que não se relacionam dentro de uma classe, examine melhor essa classe. Pode ser interessante dividi-la em mais classes, com os atributos mais relacionados juntos.

Aproveite a identificação dos atributos para rever as suas classes. Algumas classes podem estar fora de foco, sem função definida no sistema.

Algumas características de seleção de classes são sugeridas por Coad e Yourdon:

- A classe será útil somente se a informação sobre ela for necessária para o funcionamento do sistema;
- A classe em potencial deve ter um conjunto de operações identificáveis que podem mudar o valor de seus atributos de alguma maneira;

- Uma classe com um único atributo, provavelmente, será mais bem representada como um atributo de outra classe.

Algumas dicas para análise das descrições gramaticais de sistemas são sugeridas por Abbot:

QUADRO 1 - Dicas para análise das descrições gramaticais de sistemas

Gramática	Componente do Diagrama	Exemplo
Nome Próprio	Objeto	Alice, <u>Antonio</u> , ...
Nome comum (substantivo)	Classe	Funcionário, sala, ...
Substantivo e Adjetivo	Atributo	Nome do aluno, número da página, ...
Verbo de ação	Operação	Criar, calcular, ...
Verbo de Existência	Generalização/herança	É um tipo de, e um de, pode ser, ...
Verbo de Posse	Agregação e Composição	Consiste de, faz parte de, ...

INSTITUCIONAL, 2022.

Diagramas de interação

Descrição

Os diagramas de interação modelam o comportamento dinâmico dos objetos.

A interação pode representar:

- Chamada de uma operação;
- Envio de um sinal;
- Mensagem para criação ou destruição de objetos.

Os diagramas de interação são divididos em dois diagramas: sequência e colaboração (comunicação na UML 2.0).

São construídos a partir de cenários de casos de uso:

- É a realização de um caso de uso;
- Representam os vários caminhos para a realização de uma tarefa;
- Podem ser possibilidades de sucesso ou de insucesso;
- Identificados a partir do fluxo principal e dos fluxos alternativos dos casos de uso.

Diagrama de sequência

- Descreve como os objetos interagem e como eles se comunicam;
- Representa a troca de mensagens entre os objetos;
- Ênfase na ordem temporal que os eventos acontecem;
- Devem ser construídos por cenários de casos de uso.

Diagrama de colaboração (comunicação)

Enfatiza a organização estrutural dos objetos que participam de uma interação. É muito pouco utilizado por ser de leitura bem mais complexa que o diagrama de sequência.

Como ambos são derivados das mesmas informações de um modelo da UML (diagrama de classes), o diagrama de sequência e o diagrama de colaboração (comunicação) são semanticamente equivalentes, mas com enfoque de visualização diferente. Devido a isso, um diagrama de sequência pode ser convertido em um diagrama de colaboração (comunicação) e vice-versa. Geralmente, as ferramentas CASEs auxiliam nessa conversão, pois, elaborando um dos diagramas, o outro é gerado automaticamente.

Aplicação

É utilizado para modelar aspectos dinâmicos do sistema; visualizar, especificar, construir e documentar a dinâmica de determinados objetos; modelar determinado fluxo de controle de um caso de uso e validar o diagrama de classes.

Palavras-chave: modelo dinâmico, objeto.

FIGURA 20 - Fases de projeto - Diagrama de sequência



SERRA, 2019 [ADAPTADA].

Os diagramas de interação são um dos produtos das fases de análise e/ou projeto. Os diagramas de interação auxiliam também nas fases de implementação e de testes.

Notação

Diagrama de sequência

Preocupa-se com a ordem de tempo em que as mensagens são trocadas entre os objetos envolvidos em determinado processo, ou seja, quais condições devem ser satisfeitas e quais métodos devem ser disparados entre os objetos envolvidos e em que ordem durante um processo. Dessa forma, um diagrama de sequência, determina a ordem em que ocorrem os eventos, a ordem das mensagens enviadas e como os objetos interagem entre si dentro de um determinado processo.

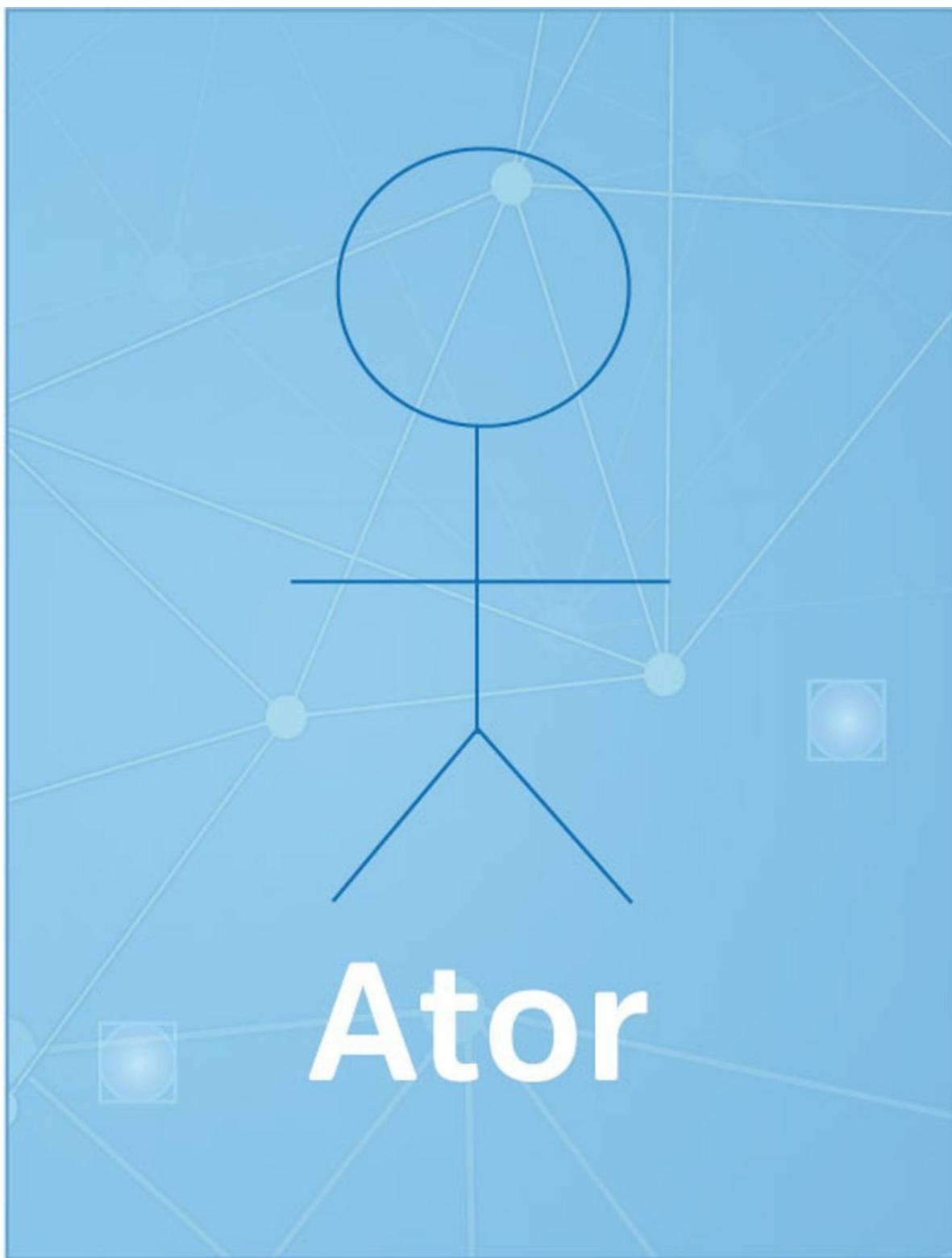
Segundo Guedes (2018), em seu livro “UML: uma abordagem prática”, normalmente, existem diversos diagramas de sequência em um projeto, um para cada processo específico do sistema. Isso exemplifica que o diagrama de sequência baseia-se no diagrama de casos de uso, contudo, o fato de existir apenas um diagrama de casos de uso não implica na existência de apenas um diagrama de sequência. O que acontece é que um diagrama de sequência, em geral, identifica-se com um determinado caso de uso e, desse modo, o diagrama de sequência também permite documentar um caso de uso.

É interessante sempre relembrar que nem todo caso de uso gera, obrigatoriamente, um diagrama de sequência, por exemplo: os casos de uso que não têm uma associação direta com um ator, sendo associado a outro caso de uso através de um <>include<>. Esses casos de uso precisam ser executados com os outros casos de uso que os utilizam e, muitas vezes, suas etapas são apresentadas no mesmo diagrama de sequência. Porém, nada impede que se defina um diagrama de sequência exclusivo para casos de uso que são utilizados por outros casos de uso por meio da associação <>include<>.

O diagrama de sequência depende, também, do diagrama de classes, uma vez que as classes dos objetos declarados no diagrama estão citadas nele.

Os elementos para a construção de um digrama de sequência são:

FIGURA 21 - Atores



INSTITUCIONAL, 2022.

Possuem a mesma descrição e conceitos presentes no diagrama de caso de uso. São entidades externas que realizam interações com o sistema através de suas funcionalidades.

FIGURA 22 - Objeto



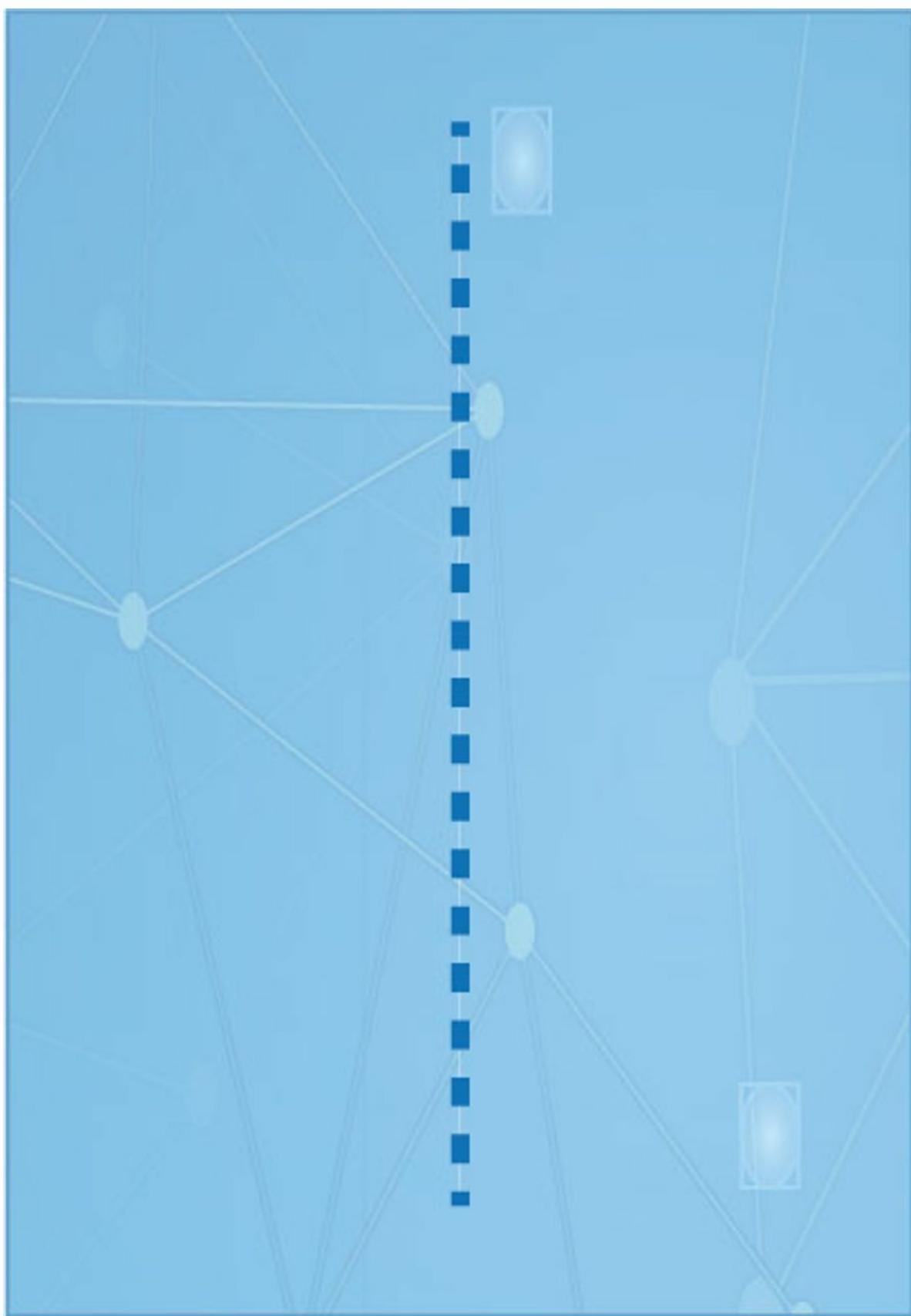
INSTITUCIONAL, 2022.

São as instâncias de uma classe, possuem nome do objeto, nome da classe e são separados por dois pontos.

Especificação da execução

Representa o período em que o objeto desempenha uma ação.

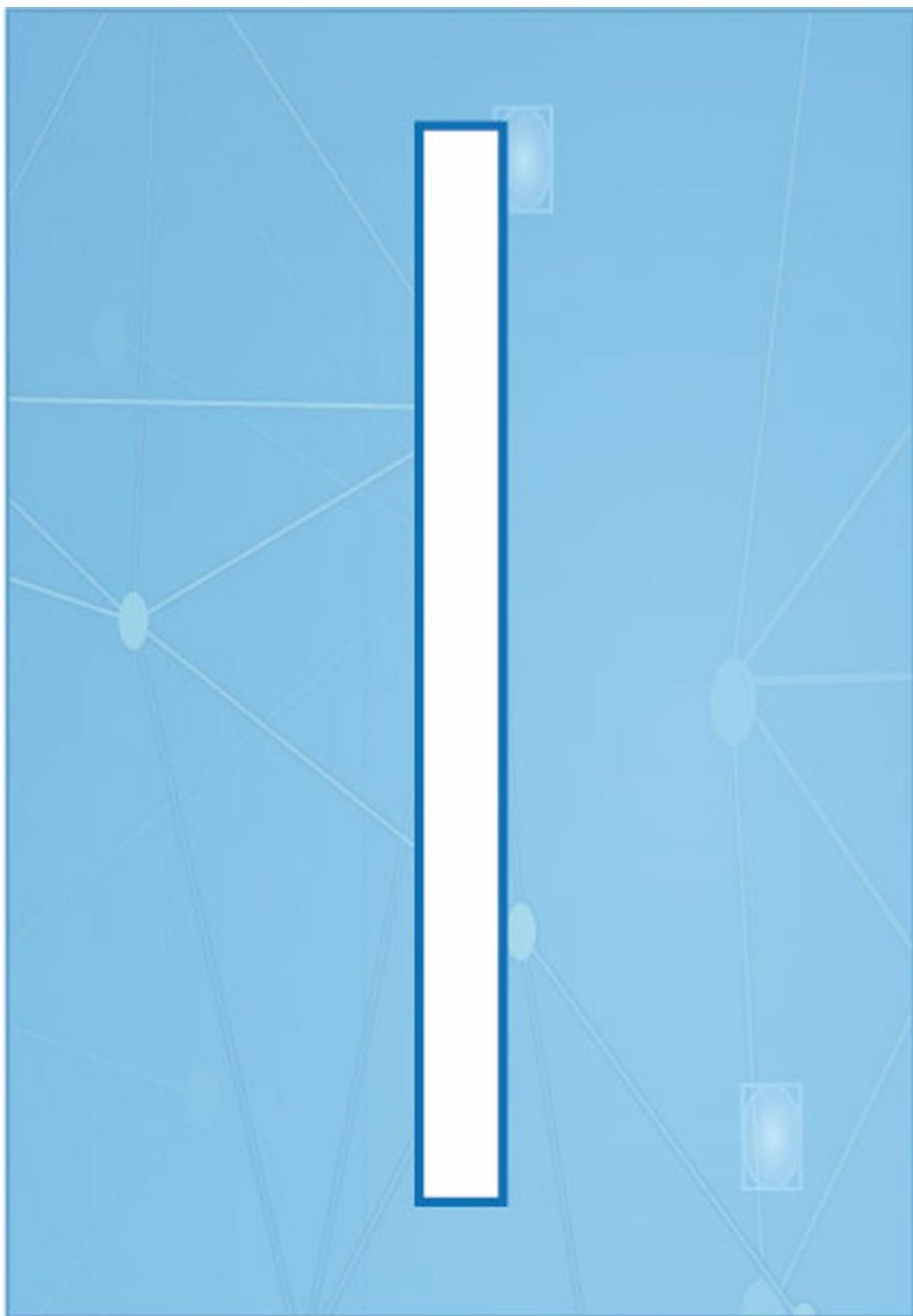
FIGURA 23 - Linha de vida



INSTITUCIONAL, 2022.

Representa a existência do objeto em um período de tempo.

FIGURA 24 - Ativação



É o estímulo recebido pelo objeto e pela ação realizada.

Mensagens

Mensagens entre os objetos.

Existem as seguintes mensagens:

Seta com ponta cheia

Representa:

- Chamada de procedimento (operação/método);
- Objeto concorrente que envia um sinal e aguarda o fim da execução de atividades aninhadas.

Seta com ponta aberta

Representa:

- Mensagem assíncrona entre os objetos.

Seta com linha interrompida

Representa:

- Retorno de uma chamada de procedimento.

Pode ser suprimido quando o retorno ocorre no fim da ativação.

Regras

Um único diagrama de interação não é capaz de captar tudo sobre os aspectos dinâmicos do sistema, por isso, utilize diversos diagramas de interação para fazer a modelagem dinâmica do sistema. Crie diversas situações (cenários) para o sistema e elabore os diagramas de interação correspondentes.

Os objetos somente podem se comunicar nos diagramas de interação caso exista um relacionamento entre as classes correspondentes no diagrama de classes.

As operações executadas pelos objetos devem estar representadas no diagrama de classes. As mensagens podem ter a assinatura dos métodos das classes. Os retornos nem sempre são exibidos, mas somente os relevantes.

Construção

Podem ser construídos para:

- Fluxo principal dos casos de uso;
- Fluxos alternativos.
- Não é necessário criar o diagrama para todos os cenários. Considere sempre os cenários mais relevantes (necessitam de detalhes);
- A troca de mensagem auxilia na identificação de novas operações para as classes envolvidas na interação.

Para cada cenário:

Tutorial

00

1

Definição

Defina o cenário que irá representar;

01

2

Identificação

Identifique os objetos que participam da interação;

02

3

Alinhamento

Coloque o objeto que inicia a interação à esquerda e os demais à direita, de acordo com sua participação na interação;

03

4

Ordenação

Coloque as mensagens que os objetos enviam e recebem na ordem crescente de tempo e de cima para baixo;

04

5

Importância

Coloque as mensagens de retorno relevantes para o cenário.

A seguir serão demonstrados alguns exemplos de diagrama de sequência:

undefined



Saiba Mais

Para saber mais, confira o vídeo 'Criando diagrama de sequência'.

Acesse: https://www.youtube.com/watch?v=ypP6HQdDxYM&t=720s&ab_channel=EstudoNaWeb

Agora que você conhece o diagrama de classes e o diagrama de sequência, realize a leitura indicada a seguir. A partir dela, você irá perceber que, ainda que tenham objetivos diferentes, esses modelos em conjunto oferecem a abstração completa da troca de mensagens entre os objetos.

Estudo Guiado

Leia os capítulos 3, 4 e 5 do livro de Martin Fowler, os quais oferecem uma visão detalhada de como modelar classes que serão programadas..

[Clique no link e leia o livro](#)

FOWLER, Martin. UML Essencial. [Digite o Local da Editora]: Grupo A, 2011. E-book. ISBN 9788560031382. Disponível em:
<https://integrada.minhabiblioteca.com.br/#/books/9788560031382/>. Acesso em: 19 set. 2022.

Diagrama de atividade

O diagrama de atividades ilustra graficamente como serão o funcionamento do software, a execução de alguma de suas partes, a atuação do sistema na realidade de negócio na qual ele está inserido.

O objetivo do diagrama de atividades é mostrar o fluxo de atividades em um único processo. O diagrama mostra como uma atividade depende da outra, sendo muito parecido com um fluxograma.

Podem especificar:

- Modelo de negócio – sistema todo;
- Detalhar um caso de uso;
- Descrever uma operação de uma classe.

Permite a construção de sistemas executáveis por meio de engenharia de produção e reversa.

Os seguintes elementos compõem o diagrama de atividades:

- Estado da atividade e estado da ação;
- Transições;
- Ramificações;
- Bifurcação e união;
- Raias.

No fluxo de controle modelado por um diagrama de atividade, as coisas acontecem. É possível calcular uma expressão que defina um conjunto de valor de um atributo ou que retorne algum valor. Os estados de ação não podem ser decompostos.

FIGURA 25 - Estado inicial



SERRA, 2019 [ADAPTADA].

Determina que o diagrama inicia em algum lugar.

- É obrigatório;
- Somente um estado inicial é permitido.

FIGURA 26 - Estado final



SERRA, 2019 [ADAPTADA].

É o estado que indica o fim do ciclo de vida de um objeto.

- É opcional;
- Pode não existir se o fluxo é infinito;
- Representação.

FIGURA 27 - Transição

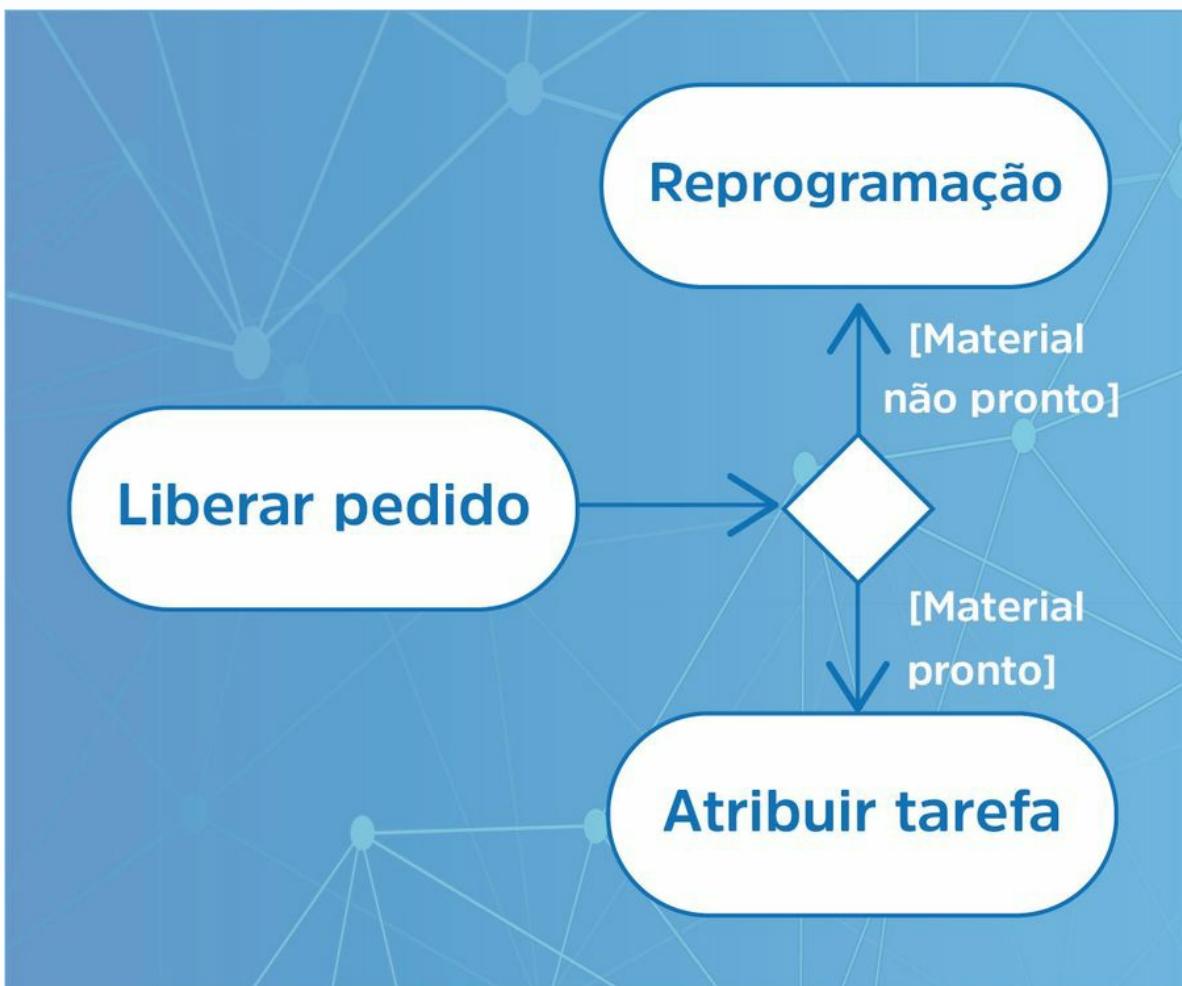




SERRA, 2019 [ADAPTADA].

- Quando uma ação ou atividade está completa, o fluxo de controle passa para a atividade seguinte;
- Esse fluxo é representado pelas transições.

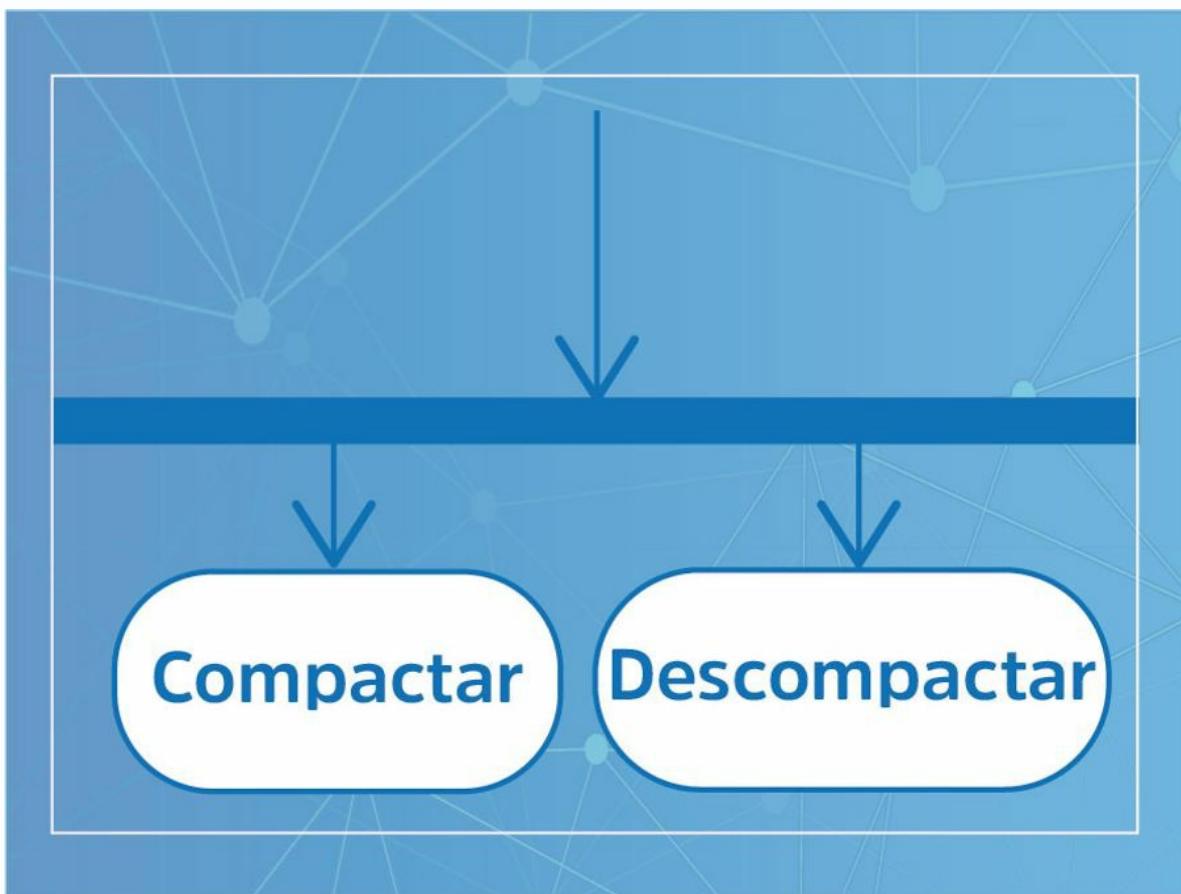
FIGURA 28 - Ramificações



SERRA, 2019 [ADAPTADA].

- São caminhos alternativos de um fluxo de controle;
- Podem ter uma transição de entrada e duas ou mais de saída;
- Em cada transição de saída, é colocada uma expressão booleana, avaliada após a entrada na ramificação.

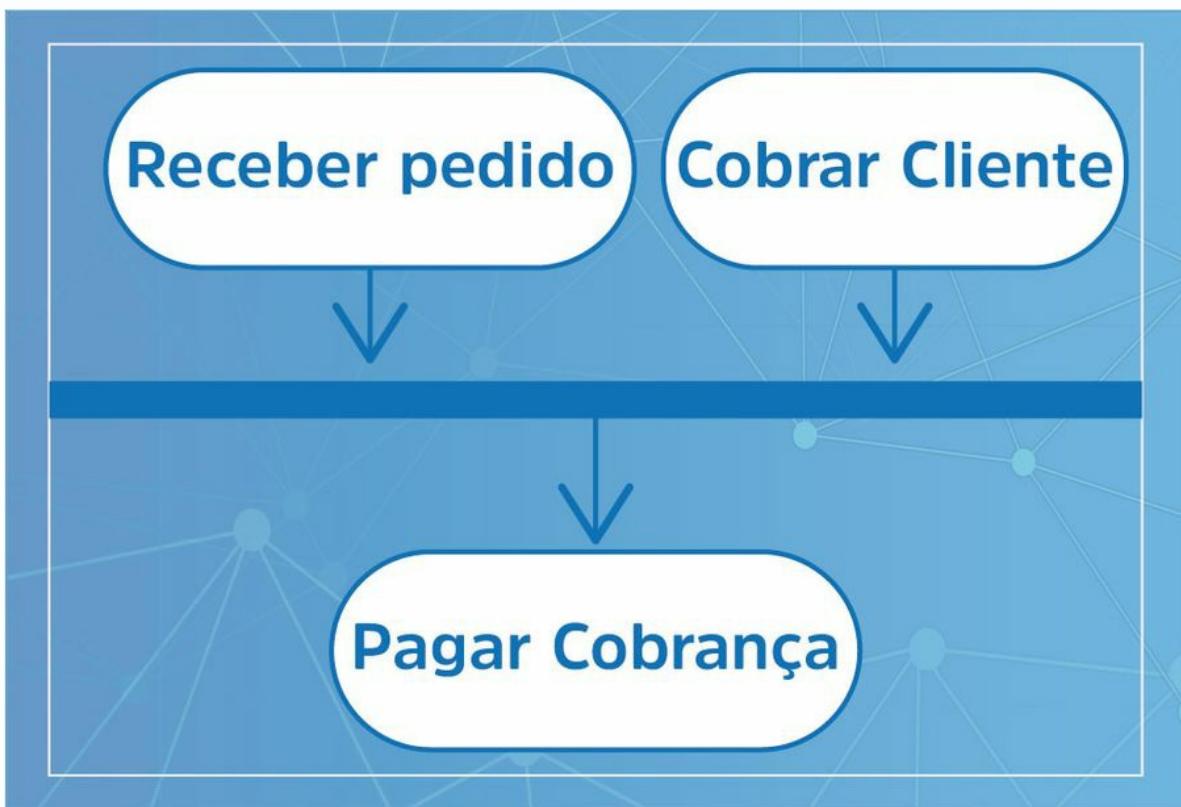
FIGURA 29 - Bifurcação



SERRA, 2019 [ADAPTADA].

- Uma bifurcação representa a divisão de um mesmo fluxo de controle em dois ou mais fluxos de controle concorrentes;
- A bifurcação poderá ter uma única transição de entrada e duas ou mais transições de saída, cada uma das quais representa um fluxo de controle independente.

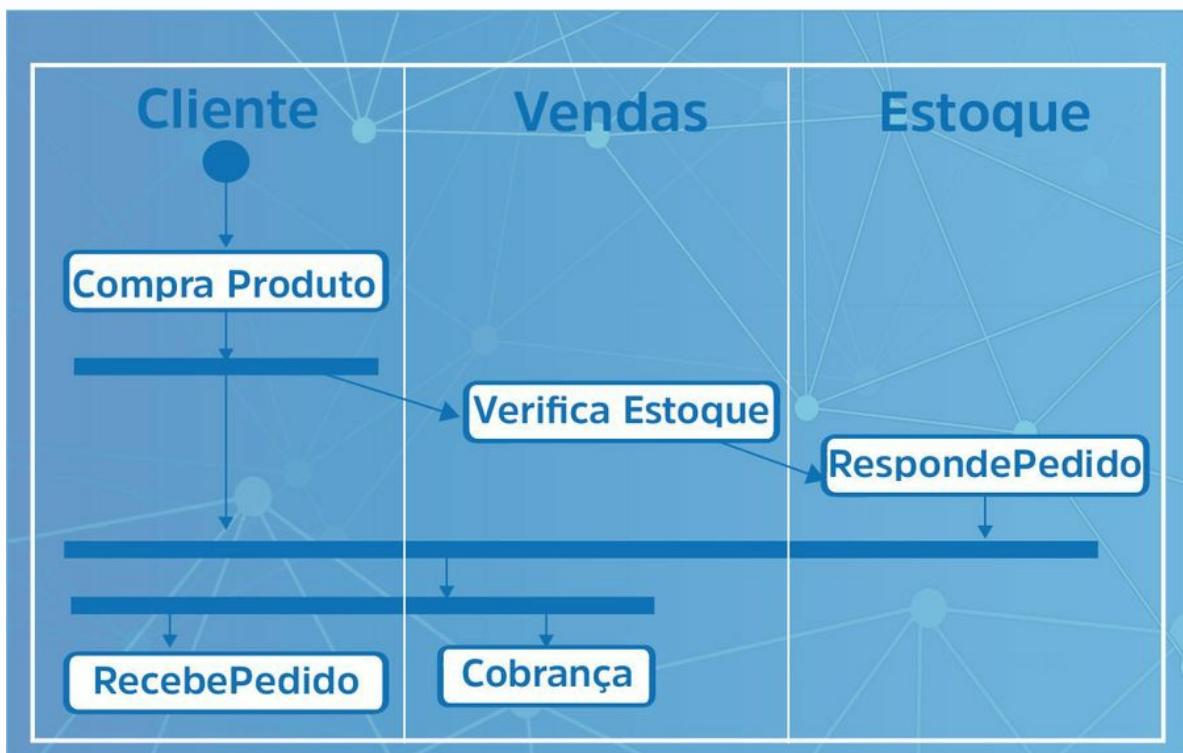
FIGURA 30 - União



SERRA, 2019 [ADAPTADA].

- Uma união representa a sincronização de dois ou mais fluxos de controle concorrentes;
- A união poderá ter duas ou mais transições de entrada e uma única transição de saída.

FIGURA 31 - Raias



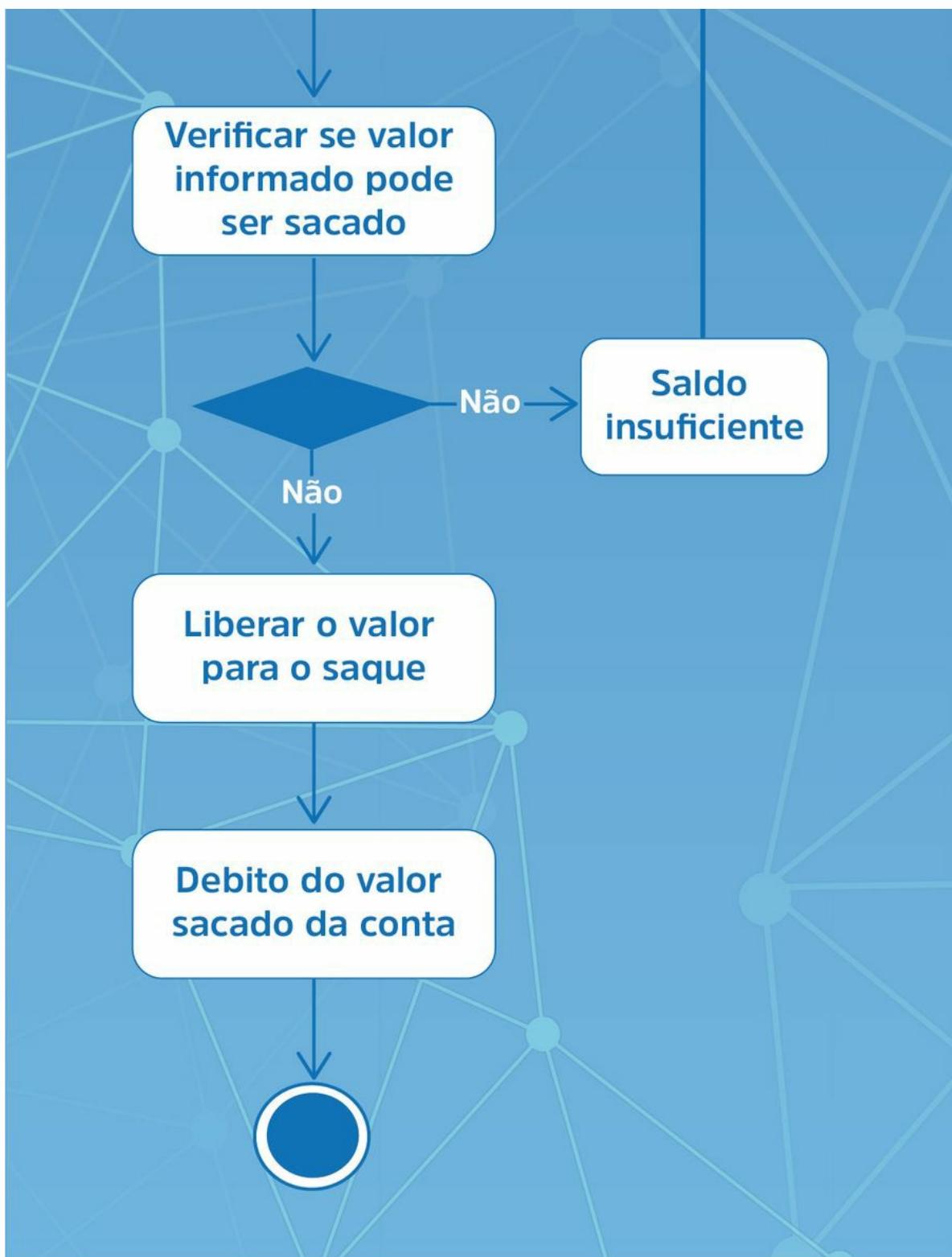
SERRA, 2019 [ADAPTADA].

- Particiona as atividades em grupos;
- Cada grupo representa um responsável pelas atividades;
- Cada grupo é colocado em uma raia e deve ter nome único;
- Muito útil no mapeamento de fluxos de negócio.

A seguir, será demonstrado um exemplo do diagrama de atividade (Realizar saque):

FIGURA 32 - Exemplo do diagrama de atividade (Realizar saque)





INSTITUCIONAL, 2022.



Saiba Mais

Para saber mais confira o vídeo 'Entenda o Diagrama de Atividade'.

Acesse: https://www.youtube.com/watch?time_continue=3&v=vReuK7_tYWc&feature=emb_logo&ab_channel=EstudoNaWeb

Chegamos ao fim deste conteúdo. Confira a seguir, uma breve síntese sobre o que vimos até aqui.

Recurso Externo

Recurso é melhor visualizado no formato interativo

Referências

Bibliográficas

GUEDES, G. T. A. **UML 2**: uma abordagem prática. 3. ed. São Paulo: Novatec, 2018.

INTRODUÇÃO a UML. **Devmedia**, 2009. Disponível em: <https://www.devmedia.com.br/introducao-a-uml/6928>. Acesso em: 12 jul. 2020.

LARMAN, C. **Utilizando UML e padrões**: uma introdução à análise e ao projeto orientados a objetos e desenvolvimento iterativo. Porto Alegre: Bookman, 2011.

MEDEIROS, E. **Desenvolvendo software com UML 2.0**: definitivo. São Paulo: Pearson Makron Books, 2004.

O QUE é UML e diagrama de caso de uso: introdução prática à UML. **Devmedia**, [2020]. Disponível em: <https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408>. Acesso em: 12 jul. 2020.

SERRA, A. P. G. **Requisitos, análise e projeto orientado a objetos**. São Paulo: 2019.