# ENGENHARIA DE REQUISITOS

**Sheila Reinehr**

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

# Diagrama de casos de uso

## Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Identificar os atores que interagem com um sistema de *software*.
- Projetar casos de uso do sistema.
- Representar os relacionamentos no diagrama de casos de uso.

## Introdução

Ao darmos início ao desenvolvimento de um *software*, a primeira questão que surge é "O que deve ser construído?". Embora a pergunta seja simples, a resposta não é. Elicitar requisitos é uma tarefa complexa, que exige tempo, planejamento e experiência. Dependendo do contexto, da complexidade do projeto e dos interesses conflitantes, identificar os requisitos sob o ponto de vista dos diversos *stakeholders* pode ser um desafio até mesmo para os analistas de requisitos mais experientes.

Uma das ferramentas que podem ser utilizadas para apoiar esse processo e representar graficamente os requisitos funcionais é o diagrama de casos de uso, que oferece uma forma simples de comunicação com os *stakeholders* em torno das funcionalidades e dos serviços que serão oferecidos aos usuários. O diagrama de casos de uso pode ser usado desde as etapas iniciais da elicitação de requisitos, como um instrumento de apoio para as entrevistas, reuniões ou *workshops*. Ele pode também apoiar o gerente de projetos como uma forma de documentação gráfica do escopo funcional.

Neste capítulo você vai estudar a elaboração de um diagrama de casos de uso, iniciando pela identificação dos atores que interagem com o *software*. Em seguida vai ver como projetar os casos de uso em si e como identificar e representar os relacionamentos dos atores com outros atores, dos atores com os casos de uso e dos casos de uso entre si.

# 1 Identificação dos atores

O **diagrama de casos de uso** faz parte do conjunto de diagramas de comportamento propostos pela UML (*Unified Modeling Language*), conforme ilustra a Figura 1.

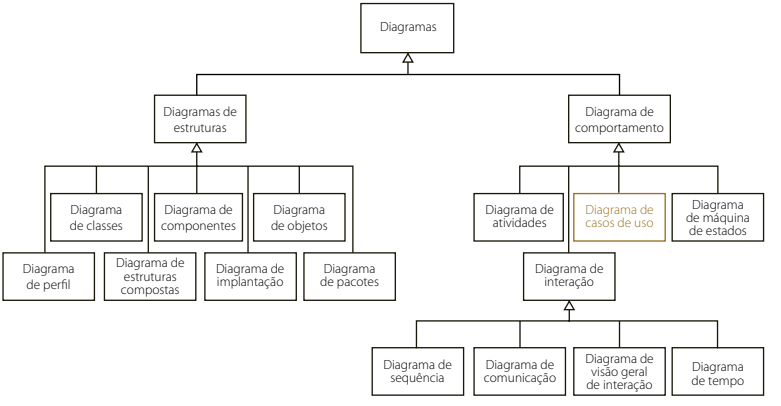


**Figura 1.** Diagramas da UML.

*Fonte:* Adaptada de Object Management Group (2017).

A primeira proposta de utilização de uma abordagem centrada no usuário com base em casos de uso foi feita por Ivar Jacobson, em 1992, que, na época, trabalhava na Erickson. Junto com James Rumbaugh e Grady Booch, Jacobson criou a linguagem de modelagem UML. Essa foi a base para outras abordagens centradas no usuário, incluindo as abordagens ágeis.

O diagrama de casos de uso representa o usuário e suas interações com o sistema, portanto ele é uma ferramenta útil quando o sistema em desenvolvimento é um sistema de informação, ou seja, quando existem diversas interações desse tipo para serem projetadas. Sistemas cuja complexidade não esteja na interação com o usuário, mas sim em seu processamento interno, não se beneficiarão desse tipo de representação (DENNIS; WIXON; ROTH, 2012).

Não existe um primeiro passo obrigatório para iniciar a construção do diagrama de casos de uso. Pode-se dar início tanto pela identificação dos próprios casos de uso quanto pela identificação dos atores que irão interagir com o *software*.

De acordo com a UML (OBJECT MANAGEMENT GROUP, 2017, documento *on-line* ), "[...] um ator especifica um papel que é desempenhado por um usuário ou qualquer outro sistema que interage com o sujeito". Entende-se por **sujeito** o sistema que está sendo construído. Um **ator** reside fora das fronteiras da aplicação, portanto ele pode ser compreendido como um papel que um usuário pode assumir ao interagir com o *software* para executar uma função. É uma espécie de chapéu que ele pode usar naquele momento da interação.

O ator pode ainda ser um *hardware* especial ou outro *software* com o qual o sistema interage e que está fora das fronteiras da aplicação. Mencionamos aqui *hardware* especial como forma de destacar que só devem ser representados os equipamentos que desempenham uma função específica e de interesse do produto. Por exemplo, não é necessário representar a impressora por meio da qual será impresso um relatório, mas pode ser de interesse representar determinado sensor que transmite dados que são interpretados pelo *software* e que desencadeiam ações.

**Saiba mais**

Raciocínio similar deve ser feito para os demais *softwares* com os quais o sistema interage. Não é necessário representar como ator o banco de dados, por exemplo, mas pode ser importante representar o sistema da operadora de cartão de crédito se o caso de uso de pagamento de uma transação precisa se comunicar com a operadora diretamente.

## Como representar os atores?

Em um diagrama de casos de uso, um ator é representado por uma figura humana simplificada, que é desenhada na forma de um boneco palito, ou *stickman,* tendo abaixo dele o nome do papel que desempenha, conforme ilustra a Figura 2.

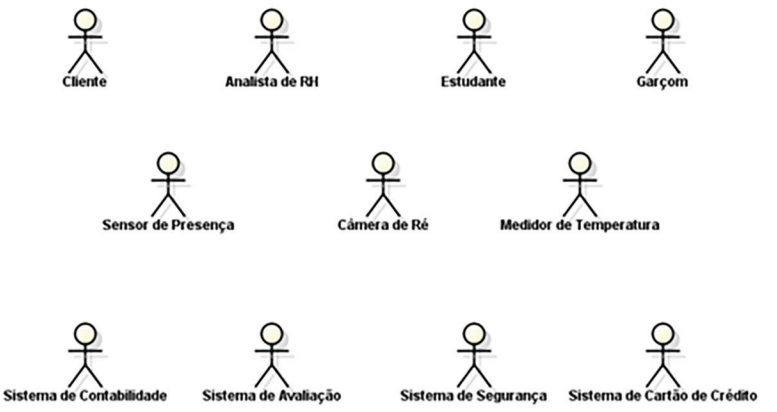


**Figura 2.** Representação dos atores em diagramas de casos de uso.

Na primeira fileira de atores da Figura 2 podemos ver exemplos de atores humanos, como Cliente, Analista de RH, Estudante e Garçom. Já na segunda fileira temos exemplos de atores que representam *hardwares* especiais, como Sensor de Presença, Câmera de Ré e Medidor de Temperatura. Na última fileira temos exemplos de atores que representam *softwares* com os quais o sistema em desenvolvimento deverá interagir*:* Sistema de Avaliação, Sistema de Segurança e Sistema de Cartão de Crédito.

Todo ator deve ter, obrigatoriamente, um nome associado a ele. Não é recomendado utilizar a denominação "Usuário", pois é muito genérica. Recomenda-se a utilização de um nome mais significativo e específico para o ator, como nos exemplos da Figura 2. Quando se tratar de um *hardware* ou outro *software*, pode-se também usar um estereótipo, indicando que se trata de um sistema. Isto é feito com a ***tag*** `<<system>>` e seu uso vai depender de como a ferramenta de modelagem UML faz essa implementação.

De acordo com a UML (OBJECT MANAGEMENT GROUP, 2017), existem ainda outras duas formas alternativas de representar os atores, mas elas praticamente não são utilizadas: o **retângulo** (como uma classe da orientação a objetos) e o **ícone** (uma figura). As ferramentas de modelagem atualmente no mercado geralmente implementam o ator como um *stickman*.

Um ator pode ser **primário** (quando dá início à ação) ou **secundário** (quando participa da ação ou apenas recebe os resultados da ação). Não existe diferença na sua representação, mas é comum que os atores primários fiquem do lado esquerdo do diagrama de casos de uso, pois é como lemos as coisas no mundo ocidental: da esquerda para a direita e de cima para baixo. Portanto, não é uma regra, mas sim uma boa prática colocá-los do lado esquerdo.

## Como identificar os atores?

Para identificar os atores de um sistema é importante iniciar identificando os usuários humanos que irão utilizar as funcionalidades. Em seguida, deve-se identificar com que dispositivos de *hardware* ou outros sistemas de *software* ele irá interagir. Algumas perguntas podem ajudar, acompanhe (GUEDES, 2018):

- Que tipos de usuários poderão utilizar o sistema?
- Quais usuários estão interessados ou utilizarão quais funcionalidades e serviços do *software*?
- Quem fornecerá informações ao sistema?
- Quem utilizará as informações do sistema?
- Quem poderá alterar ou mesmo excluir informações do sistema?
- Existe algum outro *software* que interagirá com o sistema?
- Existe algum *hardware* especial (como um robô, por exemplo) que interagirá com o *software*?

Essas questões podem ser complementadas com as que foram propostas por Wiegers e Beatty (2013), listadas a seguir:

- Quem é notificado quando alguma coisa ocorre no sistema?
- Quem provê informações ou serviços para o sistema?
- Quem o sistema ajuda a responder ou a completar uma tarefa?

Seidl *et al.* (2012) destacam ainda a importância de questionar o seguinte:

- Quem é responsável pela administração do sistema?

O primeiro contato com o patrocinador do projeto apontará para os *stakeholders* que servirão de fonte de informação para a elicitação de requisitos. Alguns desses *stakeholders* serão também usuários do sistema posteriormente

e terão um ator para os representar na especificação. Mas tenha em mente que um ator geralmente representa uma classe de usuários e que nem todos os *stakeholders* que serão fonte de informação serão usuários ou terão seus papéis associados a atores.

**Exemplo**

Um especialista em tributação pode ser uma fonte de informação para esclarecer detalhes tributários, mas pode não ser um usuário do sistema.

Desde o primeiro contato com o patrocinador, o analista de requisitos deve tentar identificar as classes de usuários (atores).

## 2 Projeto dos casos de uso

Um caso de uso é uma execução completa de uma funcionalidade que entrega um valor para o usuário que a utiliza. Esse conceito é muito importante para que tenhamos em mente o que realmente significa um caso de uso — ele é uma funcionalidade completa e que entrega algum valor. É comum vermos diagramas de caso de uso com diversos casos de uso que não são casos de uso em si, mas sim passos de um caso de uso.

**Fique atento**

Fique atento ao conceito de caso de uso! Quando estiver projetando um diagrama de casos de uso e perceber que, para completar uma funcionalidade, precisa de diversos casos de uso, procure identificar se você não está transformando indevidamente uma etapa do caso de uso em outro caso de uso.

Outra situação é quando determinada etapa se repete em diversos casos de uso e ela pode ser agrupada como um caso de uso que é chamado a partir dos demais em uma relação de inclusão (*include*).

**Exemplo**

Na dúvida, lembre-se do exemplo do caixa eletrônico. O cliente não vai ao caixa eletrônico apenas para inserir seu cartão e ir embora, ele vai para realizar uma transação, que pode ser um saque, por exemplo. Portanto, Inserir Cartão não é um caso de uso, mas sim um passo de Retirar Dinheiro. Todos os passos necessários para que o cliente possa cumprir o seu objetivo de sacar dinheiro constituem o caso de uso Retirar Dinheiro. Isso inclui identificar-se (qualquer que seja a forma exigida pela tecnologia: senha, biometria etc.), preencher as informações necessárias à operação (valor, tipo de cédula etc.) e concluir a solicitação (sair com o dinheiro).

Já a parte de identificação do cliente pode ser agrupada em um caso de uso Identificar-se, que será chamado a partir de todos os demais casos de uso que precisem que a identificação do usuário seja validada.

## Como identificar os casos de uso?

Os casos de uso são funcionalidades que o sistema deve prover, portanto, estamos falando de requisitos funcionais. Isso não quer dizer que cada requisito funcional se transforme em um caso de uso. Não é um relacionamento um para um. Um requisito pode estar em um nível de abstração mais alto e pode se transformar em diversos casos de uso. Por outro lado, um requisito pode estar definido em uma granularidade bem pequena e não constituir um caso de uso em si, mas parte de um caso de uso. Nessa situação, um grupo de requisitos funcionais pode gerar um único caso de uso.

A identificação de casos de uso pode ser feita utilizando-se as diversas técnicas de elicitação de requisitos, como entrevista, observação, reunião, *brainstorming*, grupo focal, sessão JAD (Joint Application Design) etc. O importante é que todo o escopo do projeto ou da versão/*sprint* tenha sido coberto.

Podemos utilizar estas perguntas, que podem ajudar na identificação dos casos de uso (SEIDL *et al.*, 2012):

- Quais são as tarefas principais que um ator deverá executar?
- Um ator deseja consultar ou até mesmo modificar alguma informação que existe no sistema?
- Algum ator precisa informar o sistema sobre mudanças que ocorreram em outros sistemas?
- Algum ator precisa ser informado sobre eventos inesperados dentro do sistema?

## Como representar os casos de uso?

Um caso de uso é representado por uma elipse com seu nome dentro ou logo abaixo. O caso de uso representa uma ação e, portanto, deve ser nomeado com um verbo no infinitivo, conforme ilustra a Figura 3. Ele faz parte do sistema que está sendo definido e, portanto, fica dentro do retângulo que representa o sistema, ao contrário dos atores, que ficam do lado de fora.


**Figura 3.** Representação de casos de uso.

Quando o sistema tiver muitas funcionalidades, para que o diagrama de caso de uso não fique excessivamente poluído, pode-se optar por representar apenas os casos de uso que sejam mais importantes e que representem a razão de existir do sistema, os chamados **casos de uso primários**. Podem ser considerados casos de uso secundários os que apenas mantêm cadastros ou os que emitem relatórios/consultas simples. Estes podem ficar fora do diagrama de casos de uso quando ele for muito complexo.

Existe uma situação especial que divide a opinião dos autores relacionada às funcionalidades que são disparadas pelo próprio sistema, por meio de eventos de controle ou pelo atingimento de determinado momento no tempo — por exemplo, quando um processamento é executado diariamente de forma automática, digamos, a partir das 22h. Como representar esse caso específico no diagrama de casos de uso?

Uma forma é considerar que o próprio sistema é um ator e que ele é o responsável por iniciar as funcionalidades que devem ser disparadas por ele sob determinadas circunstâncias. Embora seja uma representação clara e compreensível, o conceito de ator é violado, pois um ator é algo externo ao sistema. Outra forma de realizar essa representação é criar um ator *scheduler*, que pode ser considerado como um "agendador" de tarefas automáticas, ao qual ficam associados os casos de uso dessa natureza.

Apenas o diagrama de casos de uso não é suficiente para expressar toda a complexidade de uma funcionalidade. É necessário que seja elaborada uma especificação de casos de uso, que vai detalhar os passos e condições de execução dos casos de uso. Essa especificação não é objeto deste capítulo e, portanto, não será tratada aqui.

## 3 Identificação dos relacionamentos

Os relacionamentos no diagrama de casos de uso são chamados de **associações**. Elas podem representar relacionamentos entre os atores e os casos de uso, entre casos de uso entre si e também entre atores entre si. Vamos ver como identificar e representar cada uma delas.

### Associação entre atores e casos de uso

Uma associação entre um ator e um caso de uso indica que o ator inicia o caso de uso (geralmente chamado de ator Principal ou ator Primário) ou que o ator recebe os resultados do caso de uso que foi iniciado por outro ator. Essa associação é representada por uma linha contínua que liga o ator ao caso de uso, significando que a informação flui nos dois sentidos. Ela também pode conter uma seta em uma das pontas para indicar o sentido do fluxo da informação ou para indicar quem inicia a comunicação.

É possível que um caso de uso tenha mais do que um ator associado. Isso ocorre quando mais do que um ator é necessário para que o caso de uso seja realizado.

A Figura 4 apresenta um exemplo parcial para um *software* de Solicitação de Transporte. O ator Passageiro pode Solicitar Transporte, Alterar Percurso e Cancelar Corrida. A informação flui nos dois sentidos. O retângulo representa as fronteiras do Sistema de Solicitação de Transporte.

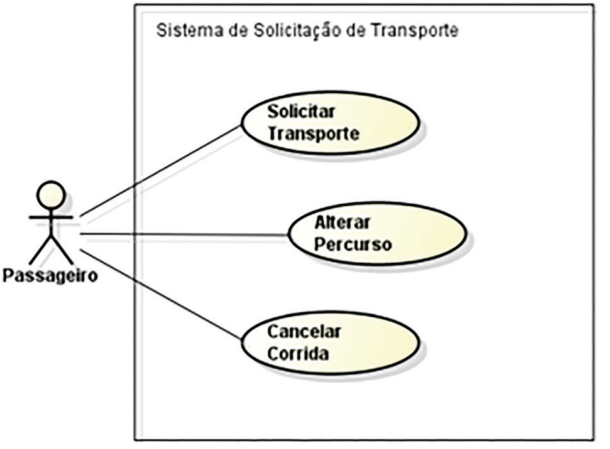


**Figura 4.** Associação entre o ator e os casos de uso.

## Associação entre casos de uso

A associação entre casos de uso pode tomar três formatos: inclusão (*include*), extensão (*extend*) e generalização (ou herança). Optamos, neste capítulo, pela manutenção do termo em inglês, pois é o mais usual, inclusive nas ferramentas que apoiam a modelagem UML.

### Relacionamento do tipo *include*

O relacionamento do tipo *include* é utilizado quando o comportamento de um caso de uso está sendo incluído dentro de outro caso de uso. Ele pressupõe que toda vez que o caso de uso que inclui for executado, o caso de uso incluído também será. O caso de uso incluído será executado de uma vez só, quando chamado pelo caso de uso que inclui. Após a sua execução, a execução dos próximos passos do caso de uso que inclui é retomada.

**Saiba mais**

Podemos entender que o relacionamento do tipo *include* foca no reuso, ou seja, na não repetição de uma mesma especificação em vários lugares diferentes, sendo similar à chamada de uma sub-rotina ou de uma função, muito comum em diversas linguagens de programação.

O *include* é representado por uma seta pontilhada que parte do caso de uso que inclui para o caso de uso que é incluído. É comum que seja também adicionado um estereótipo com a palavra `include` entre dois sinais de menor (<<) e maior (>>), o que normalmente é feito pelas ferramentas de modelagem UML de forma automática.

Vamos acompanhar o exemplo da Figura 5, que reflete uma das funcionalidades de um *site* de compras *on-line*.

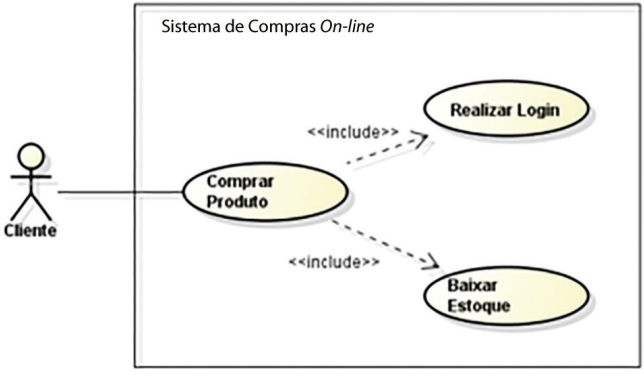


**Figura 5.** Relacionamento do tipo *include*.

O diagrama de casos de uso representado na Figura 5 diz que o ator Cliente só poderá comprar um produto se estiver logado. Note que a seta pontilhada que une os dois casos de uso aponta do caso de uso que inclui (Comprar Produto), para o caso de uso que é incluído (Realizar Login). Da mesma forma, quando o Cliente realiza a compra, será realizada a baixa no estoque (Baixar Estoque), representada pela relação *include* entre os dois casos de uso.

De acordo com as definições da UMLv2.5.1 (OBJECT MANAGEMENT GROUP, 2017, documento *on-line*),

> O relacionamento de *include* é para ser usado quando existem partes comuns do comportamento de um ou mais casos de uso. Esta parte comum é então extraída em um caso de uso separado, para ser incluída por todos os casos de uso base que tenham esta parte comum. Como o uso principal do relacionamento de *include* é para o reuso de partes comuns, o que sobra em um caso de uso base é normalmente incompleto em si, mas dependente da parte incluída para fazer sentido. Isto é refletido na direção do relacionamento, indicando que o caso de uso base depende da adição e não vice-versa.

## Relacionamento do tipo *extend*

O relacionamento do tipo *extend* é utilizado quando o comportamento de um caso de uso estende o comportamento de outro caso de uso, sob circunstâncias específicas. O relacionamento *extend* especifica como e quando o comportamento definido no caso de uso que estende pode ser inserido dentro do comportamento definido do caso de uso estendido. Essa extensão pode ocorrer em um ou mais pontos de extensão do caso de uso estendido. O *extend* é usado quando o comportamento é adicionado de forma condicional, ou seja, sob certas circunstâncias.

O *extend* é representado por seta pontilhada que parte do caso de uso que estende para o caso de uso que é estendido. É comum que seja também adicionado um estereótipo com a palavra `extend` entre dois sinais de menor (<<) e maior (>>), o que normalmente é feito pelas ferramentas de modelagem UML de forma automática.

Vamos acompanhar o exemplo da Figura 6, que representa as formas de pagamento que o ator Cliente tem ao realizar uma compra *on-line*. O Cliente, ao pagar a compra, poderá optar pelo pagamento na forma de boleto bancário ou por meio do cartão de crédito. O retângulo externo representa as fronteiras do Sistema de Compras *On-line*.

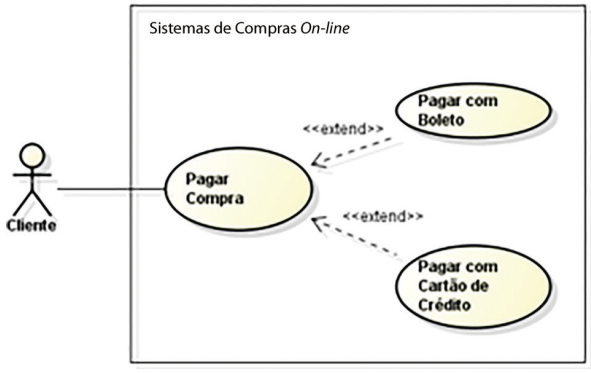


**Figura 6.** Relacionamento do tipo *extend*.

É possível ainda que seja identificado explicitamente no diagrama quais são os pontos de extensão, conforme ilustrado na Figura 7. Nesse caso, essa identificação fica dentro da elipse que representa o caso de uso, com a expressão *extension points* (pontos de extensão) logo abaixo do nome do caso de uso, em um compartimento separado por uma linha contínua.

Isso significa que, se essa condição for satisfeita, o caso de uso que estende será incorporado. Ao identificar que o Cliente selecionou a opção Boleto, o caso de uso Pagar Compra é estendido pelo caso de uso Pagar com Boleto. Analogamente, caso o Cliente tenha selecionado a opção Cartão de Crédito, o caso de uso Pagar Compra será estendido pelo caso de uso Pagar com Cartão de Crédito.

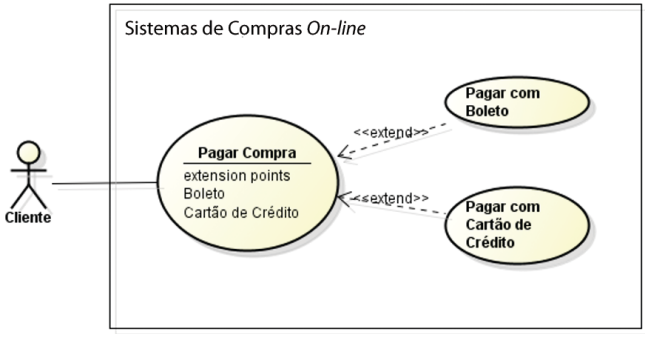


**Figura 7.** Relacionamento do tipo *extend* com pontos de extensão explícitos.

Um caso de uso pode ser utilizado para estender mais de um caso de uso. Da mesma forma, um caso de uso estendido pode também conter uma extensão. Em cada ponto de extensão apenas uma das opções de extensão será executada, ou seja, como se pode observar na Figura 7, ou o Cliente paga com Boleto ou o Cliente paga com Cartão de Crédito.

De acordo com a definição da UMLv2.5.1 (OBJECT MANAGEMENT GROUP, 2017, documento *on-line*),

> O caso de uso estendido é definido de forma independente do caso de uso que estende e faz sentido independentemente do caso de uso que estende. Por outro lado, o caso de uso que estende tipicamente define um comportamento que pode não necessariamente fazer sentido sozinho. Ao invés disto, o caso de uso que estende define um conjunto de incrementos de comportamento modulares que ampliam uma execução de um caso de uso estendido sob determinadas condições.

## Relacionamento do tipo generalização

O relacionamento do tipo generalização funciona como o relacionamento de herança da orientação a objetos, ou seja, é utilizado quando o comportamento de um caso de uso (chamado de caso de uso geral) é herdado por outro caso de uso (chamado de caso de uso especializado). Os casos de uso especializados herdam todos os relacionamentos do caso de uso geral ao qual estão associados, mas a recíproca não é verdadeira.

O relacionamento de generalização é representado por uma linha contínua que inicia no caso de uso especializado e termina em uma seta aberta que aponta para o caso de uso geral. A Figura 8 apresenta um exemplo de generalização entre casos de uso em um exemplo parcial de um Sistema Bancário. Conforme se pode observar, um Cliente pode abrir uma Conta em um banco que pode ser uma Conta Corrente ou uma Conta Poupança. O comportamento do caso de uso Abrir Conta é especializado de acordo com o tipo da conta, executando os casos de uso Abrir Conta Corrente e Abrir Conta Poupança, conforme a seleção do Cliente.

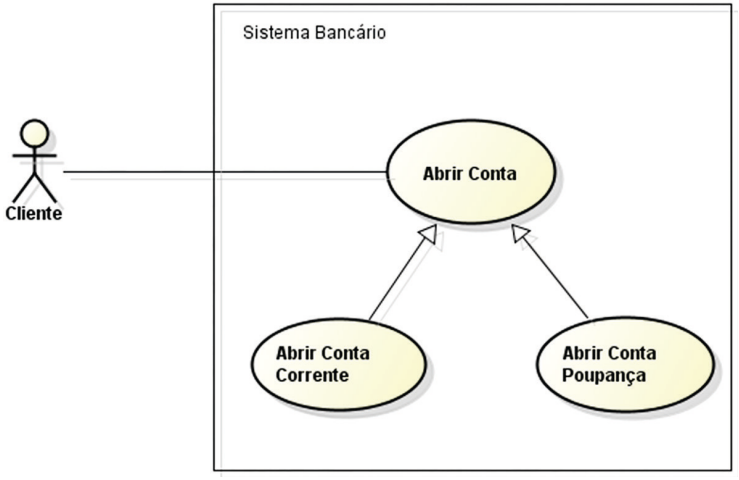


**Figura 8.** Relacionamento do tipo generalização de casos de uso.

O relacionamento do tipo generalização também é aplicável para os atores, e o conceito é exatamente o mesmo. Um ator especializado herda todos os casos de uso e os relacionamentos do ator geral.

Um exemplo pode ser visto na Figura 9, que mostra parcialmente um Sistema de Compras *On-line*. Um Visitante pode realizar o caso de uso Pesquisar Produtos, mas não pode executar o caso de uso Comprar Produtos. Ele só poderá Comprar Produtos quando se tornar um Cliente, que é um usuário cadastrado e logado. Por sua vez, um Cliente poderá tanto Pesquisar Produtos quanto Comprar Produtos.

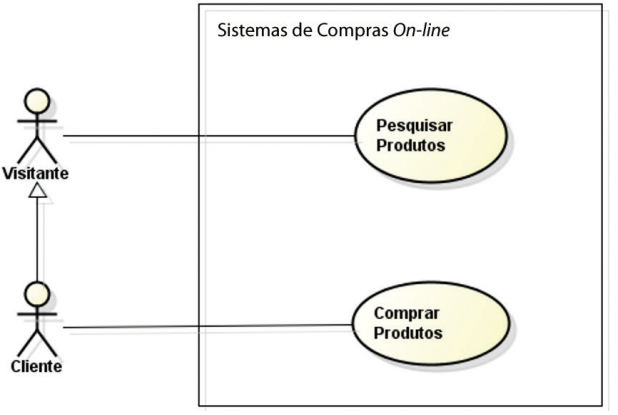


**Figura 9.** Relacionamento tipo generalização de atores.

O SWEBOK (*Software Engineering Body of Knowledge*) v3 (BOURQUE; FARLEY, 2014, p. 1–14) alerta o seguinte sobre os requisitos:

> Nem todas as organizações possuem a cultura de documentar e gerenciar os requisitos. É muito comum em empresas dinâmicas como *startups*, direcionadas por uma forte visão orientada a produto e recursos limitados, ver a documentação dos requisitos como um *overhead* desnecessário. Na maioria das vezes, no entanto, na medida em que estas empresas expandem, a sua base de clientes cresce, e o seu produto começa a evoluir, elas descobrem que necessitam recuperar os requisitos que motivaram as funcionalidades do produto de forma a avaliar o impacto das mudanças propostas. Consequentemente, a documentação dos requisitos e o gerenciamento de mudanças são a chave para o sucesso de qualquer processo de requisitos.

## Referências

BOURQUE, P.; FAIRLEY, R. *SWEBOK:* guide to the *software* engineering body of knowledge version 3. [*S. l.*]: IEEE Computer Society, 2014.

DENNIS, A.; WIXON, B.; ROTH, R. *System analysis and design*. 5. ed. Hoboken: John Wiley and Sons, 2012.

GUEDES, G. *UML2*: uma abordagem prática. 3. ed. São Paulo: Novatec, 2018.

OBJECT MANAGEMENT GROUP. *Unified Modeling Language®*: OMG UML® v2.5.1. [*S. l.*], 2017. Disponível em: https://www.omg.org/spec/UML/About-UML/. Acesso em: 2 fev. 2020.

SEIDL, M. *et al*. *UML@Classroom:* an introduction to object-oriented modeling. Cham: Springer, 2012.

WIEGERS, K. E.; BEATTY, J. *Software requirements*. 3. ed. Redmond: Microsoft Press, 2013.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS