

Caso de Uso – Include, Extend e Generalização

Published by [Plínio Ventura](#) on December 28, 2014

Caso de Uso e Programação

Fazer um Caso de Uso, dependendo do ponto de vista, não é algo muito diferente do que programar. É possível fazer um bom trabalho, sob um mesmo ponto de vista, tanto na modelagem quanto na codificação.

Sabia que é possível utilizar uma linguagem de programação Orientada a Objetos e fazer um software com programação **não orientada à objetos**?

Sim, é possível fazer em C# ou Java, por exemplo, um software programado de maneira “quase” [estruturada](#). Muito disso é [POG](#)!

Nessa mesma linha de raciocínio, é possível utilizar modelagem de caso de uso em um projeto, mas no fim das contas, ter algo mais próximo de [diagramas de fluxo de dados](#) do que de diagramas de Caso de Uso.

Obs.: infelizmente na área de software muitos profissionais **dão pouca importância à qualidade**, não se apegam os detalhes. Fazem as coisas por fazer, sem saber a fundo o que estão fazendo, ou porque estão fazendo.

Nem sempre é culpa do profissional, é uma área com muitos dirigentes despreparados.

Os diagramas de Caso de Uso são relevantes?

Relevante é. Mas **depende da qualidade** do que foi produzido.

Sempre haverá o profissional arrogante que vai analisar o diagrama de caso de uso e diz: “perdeu tempo fazendo isso? Um desenho com bonecos de palito, bolinhas e setinhas ligando as coisas?”.

Outro alguém pode falar: “para que especificar. Até minha mãe faz um diagrama melhor... desenhar bonecos e bolas não tem sentido, não agrega nada ao projeto!”.

/ Já ouvi um gerente sênior falando da mãe dele, como foi descrito. Acontece... */*

Excetuando a ironia e falta de gentileza, realmente, fazer um “desenho” (diagrama) com bonecos de palito e bolas, ligando estas coisas, sem ter [sentido semântico](#) algum nisso, com **baixa qualidade** no material produzido, não tem utilidade mesmo.

É perder tempo, tempo que poderia ser empregado em coisas mais úteis ao projeto.

Mas se for um **trabalho bem feito**, se for um diagrama **produzido com qualidade**, que realmente explora as possibilidades da técnica de modelagem de caso de uso, utiliza a técnica corretamente e **ajuda** a toda a equipe a entender e implementar o escopo do projeto da melhor forma, aí gera valor, aí **torna-se relevante**.

Relacionamento entre Casos de Uso

Relacionamentos entre Casos de Uso, principalmente para os profissionais que estão tendo o primeiro contato com o assunto, quase sempre geram alguma confusão. É natural.

As dúvidas sobre **Inclusão** (Include), **Extensão** (Extend) e **Generalização** [ou Herança] (Generalization) ocorrem com frequência, são comuns.

Existem alguns profissionais que defendem que “isso é bobagem, não precisa, include só resolve”, mas isso é como usar uma linguagem orientada a objetos mas programar o software no paradigma “procedural”; compila e executa, mas fica **um monte de benefícios para trás**, além do que, depois que a modelagem acaba fica a confusão de entender “para que serve aquilo que eu fiz”.

/ No contexto do parágrafo acima, caso você seja um profissional que se preocupa com qualidade no que faz, recomendo muito que [veja este meu vídeo](#) sobre “[fazer certo da primeira vez!](#)” */*

Vamos ao que significa cada um dos três tipos de relacionamento citados. Consideremos que temos três Casos de Uso – A, B e C, e com base nos três vamos descrever cada um dos relacionamentos.

Include

Quando o caso de uso **A** “inclui” o caso de uso **B**, significa que **sempre** que o caso de uso **A** for executado o caso de uso **B** também

será executado. A direção do relacionamento é do caso de uso que está **incluindo** para o caso de uso **incluído**.

Extend

Quando o caso de uso **B** estende o caso de uso **A**, significa que quando o caso de uso **A** for executado o caso de uso **B** poderá (poderá – talvez não seja) ser executado também. A direção do relacionamento é do caso de uso **extensor** (aqui o caso de uso B) para o caso de uso **estendido** (aqui o caso de uso A).

Generalization

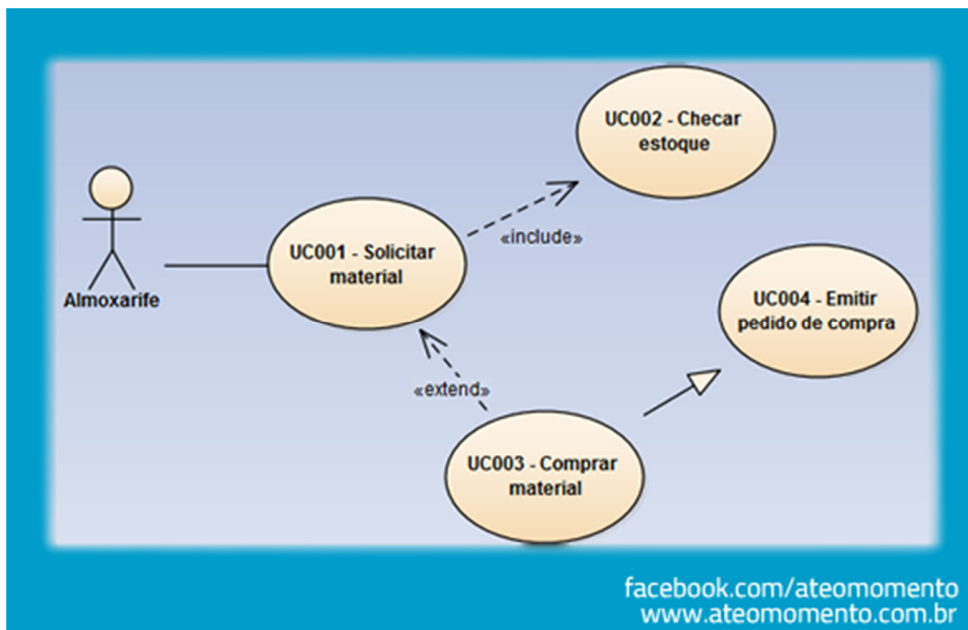
Quando o caso de uso **B** generaliza o caso de uso **C** isso significa que, além de fazer tudo que nele está especificado (ele = B), ele **também executará tudo que está especificado** no caso de uso **C**.

Muitos profissionais falam que isso não deve ser compreendido como a herança da orientação a objetos, mas na minha opinião deve ser sim, apenas (em tempo de modelagem de caso de uso) estamos num nível de abstração diferente, mas o produto final desta modelagem será software codificado.

A direção do relacionamento é sempre do **generalizador** (aqui o caso de uso B) para o **generalizado** (caso de uso C).

Exemplificando

Abaixo um diagrama com um cenário semelhante ao utilizado acima, ilustrando os relacionamentos.



No diagrama temos quatro Casos de Uso, e três relacionamentos diferentes: Include, Extend e Generalization.

Explicando o Include

O caso de uso “Solicitar Material” faz include no caso de uso “Checar Estoque”. Isso se dá porque **sempre** que houver a solicitação de material **sempre** haverá a consulta ao estoque para saber se o material está disponível.

Se sempre haverá, o relacionamento correto é o include.

Explicando o Extend

O caso de uso “Comprar Material” estende o caso de uso “Solicitar Material”. Isso se dá porque quando houver a solicitação de material, **caso o material não exista em estoque** (após consulta via o caso de uso “Checar estoque”) **poderá** ser solicitado a compra do item.

Mas também poderá não ser solicitada a compra, pois o item pode existir em estoque. Se **poderá** ser solicitada a compra (e não **sempre** será solicitada a compra) o relacionamento correto é o extend.

Explicando o Generalization

O caso de uso “Comprar Material” generaliza o caso de uso “Emitir pedido de compra”. Isso se dá porque no caso de uso “Emitir pedido de compra” **existee** especificação de como se realiza o pedido de compra, **processo que não se dá somente no contexto do almoxarifado**, mas é o mesmo em qualquer área do negócio.

Dessa forma, não justifica-se duplicar a especificação pertinente em outro caso de uso, basta **reaproveitar** o que já está pronto mas generalizado a ponto de poder ser aproveitado por alguém que o especialize.

A importância do uso correto nos relacionamentos

Especificações são feitas para serem **interpretadas**, e com base na interpretação, viabilizar a produção de software executável.

Quanto **mais qualidade** houver na especificação, **mais fácil** será de entendê-la, e **mais correta** será a interpretação de quem utilizá-la.

Essa facilidade gera **velocidade** na produção dos outros modelos (incluindo o modelo de código fonte, casos de teste etc.), **diminui a quantidade de defeitos** em potencial (quanto mais clara

uma especificação, menor a chance dela ser interpretada de forma errada) e gera outros benefícios diversos.

Concluindo

A clareza e corretude nos relacionamentos entre os casos de uso influencia diretamente na qualidade do projeto.

Muitos profissionais acham que o diagrama serve apenas para “colar as bolinhas” para que alguém os identifique e consiga “ver o que tem dentro”, ou seja, ver os cenários do Caso de Uso. Vimos que vai muito além disso...