COMO A LINGUAGEM UML PODE TE AJUDAR A DESENVOLVER MELHOR SUA APLICAÇÃO

LinkedIn Article
Leandro Teodoro
25/08/2023

O QUE É A LINGUAGEM UML E PORQUE MODELAR?

A sigla UML significa Unified Modeling Language e foi criada para ser uma linguagem gráfica de modelagem padrão para estrutura de projeto de software. Essa linguagem pode ser usada para especificar classes, funções, estruturas de bancos de dados e até design de hardware. É importante frisar que essa linguagem pode ser aplicável a outros projetos além de sistemas orientados a objetos. A UML surgiu pela junção de duas linguagens de modelagem: Booch e OMT.

É importante lembrar que a UML não é uma metodologia de desenvolvimento, assim sendo, não dita qual é a ordem de utilização dos diagramas que veremos a seguir. Entretanto, uma boa prática seria iniciar pelos casos de uso, após isso o de atividades e depois partir para os outros. Porém, isso não pode ser encarado como algo rígido.

Mas o que é um modelo? Um **modelo é uma simplificação da realidade**, ou seja, uma abstração menos complexa do mundo real. Você pode estar se perguntando porque criar um modelo, pois seria mais prático ir direto para o projeto em si. O que acontece é, criar um modelo vai lhe proporcionar muitas vantagens, como: gerenciar a complexidade, entender os riscos associados ao projeto, testar conceitos, escolher melhor as variáveis, estudar comportamentos, melhor entendimento do negócio e das regras do projeto.

A modelagem permite uma melhor compreensão do sistema a ser desenvolvido. Não é a toa que os engenheiros civis criam plantas antes de realizar uma construção. Essa modelagem permite o estudo, por exemplo, da resistência dos materiais envolvidos. Podemos ver que em alguns casos não realizar um modelo antecipado pode resultar numa catástrofe.

Todos sabemos que o tempo perdido possui um custo alto. Quem não já ficou horas tentando resolver um erro oculto no código de programação, não estou falando dos erros os quais o compilador detecta ou mesmo informa uma mensagem de “warning” e sim aquele erro que causa mau funcionamento no sistema de modo geral. Assim, modelar ajuda em uma melhor compreensão da lógica das funções e operações envolvidas. A análise do modelo responde a pergunta **“o que deve ser feito?”**

Está sendo útil para você? Então deixe seu gostei, vamos compartilhar conhecimento e ajudar outras pessoas.
Ok, vamos continuar...

TIPOS DE DIAGRAMAS UML.

Agora, vamos conhecer os principais diagramas UML que vão auxiliá-lo no seu projeto. É importante lembrar que existem outros diagramas, veremos os mais utilizados. Os diagramas são divididos em dois grupos: estruturais e comportamentais. Sendo que os estruturais descrevem aspectos estáticos do sistema como classes, interfaces e componentes. Já os comportamentais descrevem aspectos dinâmicos do sistema como fluxo de mensagens, movimentação de pacotes e alterações dos estados internos dos objetos.

Existem várias ferramentas para criação de diagramas UML, por exemplo o ArgoUML ou no Linux você pode instalar o Dia com o comando:

```
$sudo apt-get install dia
```

O primeiro será o diagrama de casos de uso, esse descreve uma sequência de ações que o sistema executa para fornecer um resultado observável. Os casos de uso são representados por uma elipse, e seus atores, representados por um boneco. Cada elipse possui o nome da ação que o ator desempenha dado o cenário. O ator não precisa ser necessariamente uma pessoa, podendo ser também outro programa. Dessa forma, esse diagrama ilustra os aspectos dinâmicos do sistema e ações indispensáveis ligadas as funcionalidades do software. Geralmente, é o primeiro diagrama criado, pois implementa aspectos fundamentais da engenharia de requisitos, identificando os casos que seu cliente espera que o programa faça.

Fonte: Do Autor

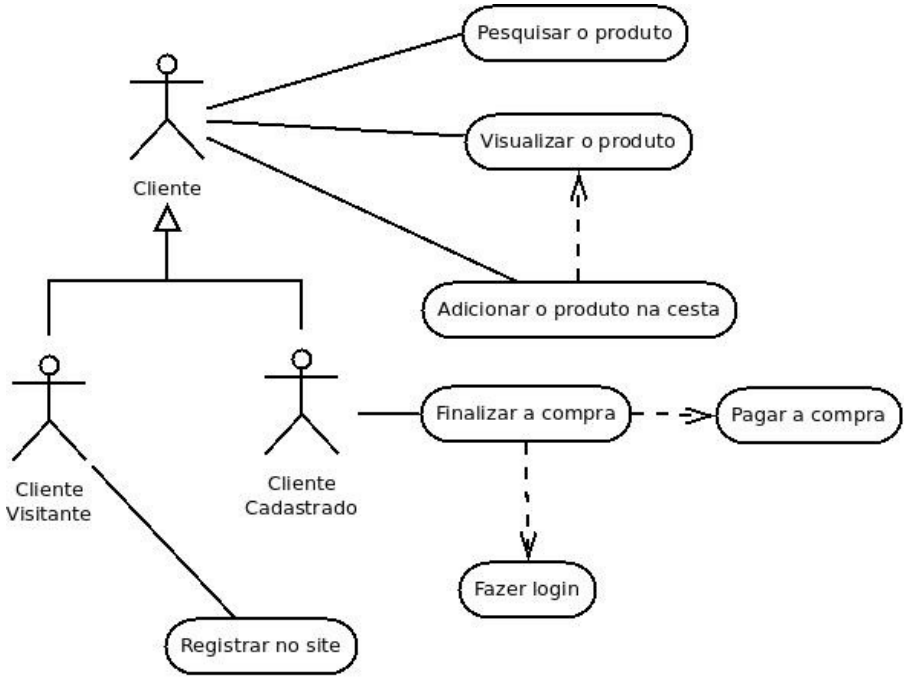

Figura 1- Diagrama de casos de uso

Alguns tipos de relacionamentos comuns entre casos de uso:

- Incluir(Include): Quando um caso de uso A é executado o caso de uso incluído B também é executado, impreterivelmente. É o caso de “finalizar a compra” onde é necessário “fazer login”, seguindo o sentido da seta tracejada.
- Estender(Extends): Quando o caso de uso A é executado, o caso de uso estendido B **poderá ou não** ser executado. É o caso de “visualizar produto” onde pode-se ou não ser executado “adicionar produto na cesta”. Observe a leitura na direção inversa a seta.
- Generalizar/Herança: Quando o caso de uso A é executado, ele pode ser entendido como uma fração específica de um caso de uso genérico que será executado também.

O segundo diagrama que iremos conhecer é o de classes, as quais são abstrações para representar objetos com características e comportamentos comuns. As classes possuem um nome, atributos, métodos e relacionamentos entre si, como: herança, associação, agregação, composição e dependência. O diagrama de classe é um dos mais importantes, pois com ele é possível observar como os objetos interagem entre si, além de apresentar uma noção ampla de como o software está estruturado.

Fonte: Martin Fowler – UML Distilled

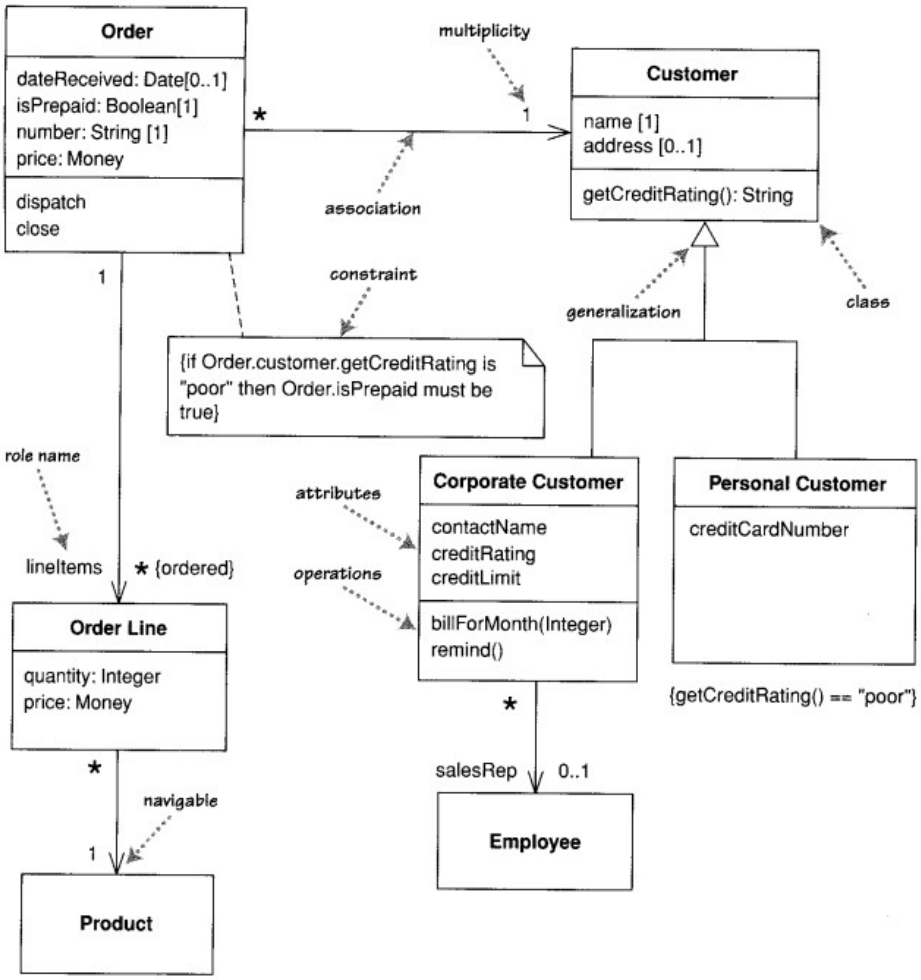

Figura 2 - Diagrama de classes

Agora vamos entender a finalidade desses relacionamentos entre classes.

- Herança(Generalization): A classe filha herda as características da classe mãe, atributos e métodos. No diagrama acima, a ponta da seta fechada e sem preenchimento identifica a classe mãe.
- Associação: Relacionamento estrutural que descreve um conjunto de vínculos, em que o vínculo é uma conexão entre objetos. Indicada pela seta ou por uma linha sólida se a associação for bidirecional.

Ainda temos outros tipos de relacionamento entre classes...

- Agregação: Tipo de relacionamento com características todo-parte. A existência do objeto parte faz sentido mesmo se o objeto todo deixar de existir. É identificado pela linha com losango fechado e sem preenchimento.

- Composição: Também é um relacionamento todo-parte, mas se o objeto todo deixar de existir as partes não fazem mais sentido, devido ao alto grau de coesão entre o todo e as partes. Esse relacionamento é identificado pelo losango preenchido.
- Dependência: Quando uma classe usa a outra, de forma que uma alteração na classe independente irá afetar a o funcionamento da classe dependente. O símbolo é a seta tracejada da classe dependente para a independente.

O último diagrama é o de atividades, o qual mostra o **fluxo** de controle de uma atividade para a outra. Esse diagrama apresenta uma visão dinâmica do processo. O diagrama de atividades lembra muito o fluxograma, então sua implementação é quase que intuitiva. As setas ditam a sequência onde ocorrem as atividades, sendo que as atividades podem ocorrer de forma sequencial ou em paralelo.

Fonte: Do Autor

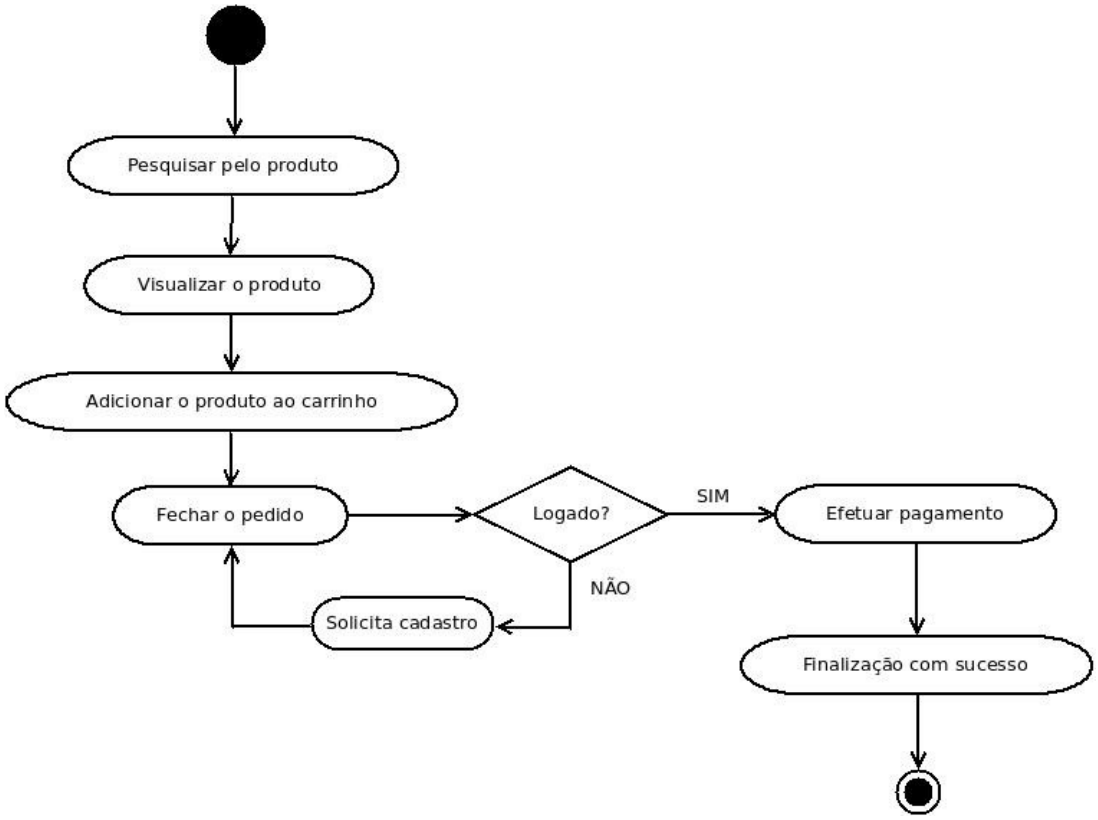

Figura 3 - Diagrama de atividades

O texto ficou extenso? Então vamos finalizar por aqui, não é o propósito detalhar uma convenção sobre UML agora, mas se quiser entender mais sobre o assunto procure os livros do Martin Fowler. Compartilhe essa informação para que possamos aprender uns com os outros, será uma ótima experiência.
Agora você já sabe, antes de inciar seu projeto de fato, seja ele qual for, tente criar um modelo. Faça testes, altere os componentes, estude os resultados. Tenho certeza que poupará muito tempo e investimento.