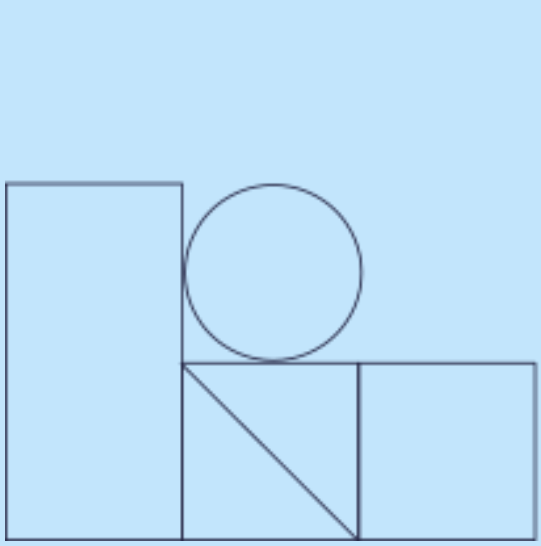
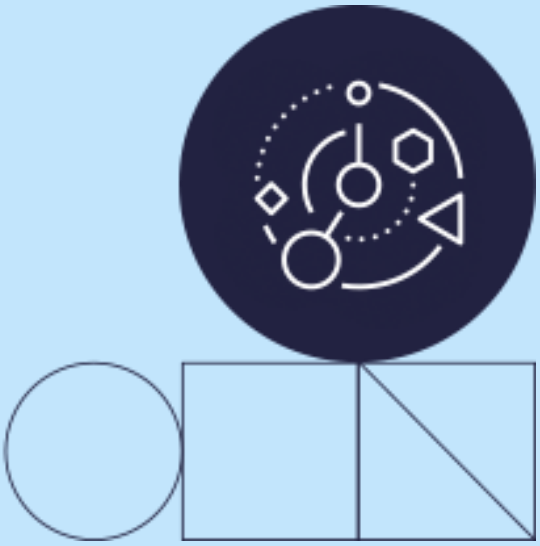
IBM SkillsBuild

IBM

OOP com Python

O que é UML e como é usado para OOP?





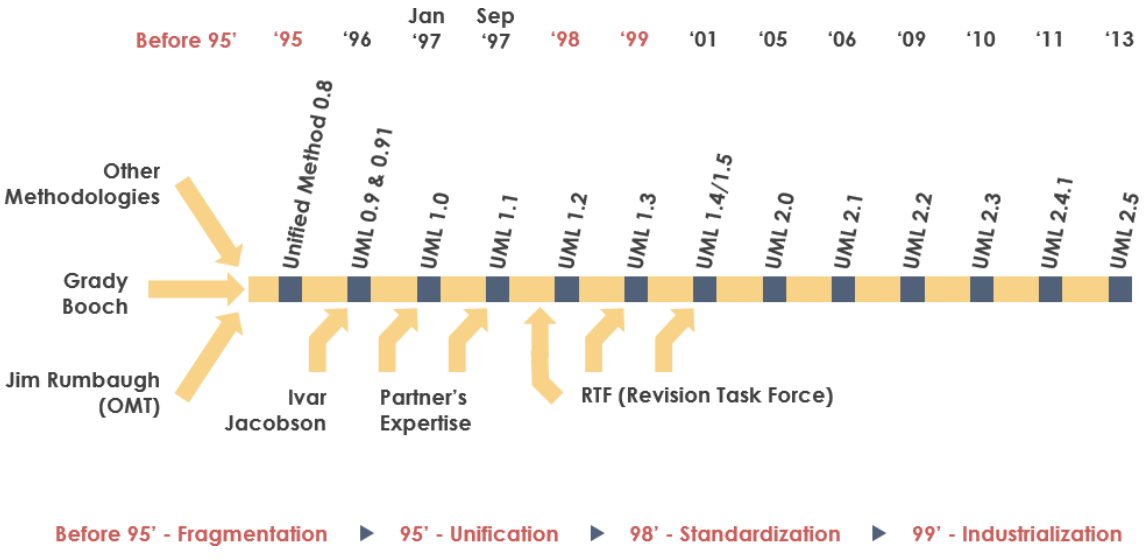
Índice

Introdução

A linguagem unificada de modelagem (UML) surgiu pela primeira vez nos anos 90 como um esforço para selecionar os melhores elementos dos muitos sistemas de modelagem propostos naquela época e combiná-los em uma única notação coerente. Desde então, tornou-se o padrão da indústria para modelagem e design de software, bem como para modelagem de outros processos no mundo científico e empresarial.

O UML é uma ferramenta para especificar sistemas de software. Tipos padronizados de diagramas nos ajudam a descrever e mapear visualmente o projeto e a estrutura de um sistema de software. Usando UML é possível modelar quase qualquer tipo de aplicação, tanto especificamente quanto independentemente de uma plataforma alvo. Embora a UML seja naturalmente orientada para a programação orientada a objetos, é igualmente fácil modelar linguagens de procedimento como C, Visual Basic, Fortran, etc.

Usar UML como uma ferramenta para definir a estrutura de um sistema é uma maneira muito útil para gerenciar sistemas grandes e complexos. Ter uma estrutura claramente visível facilita a introdução de novas pessoas em um projeto existente.



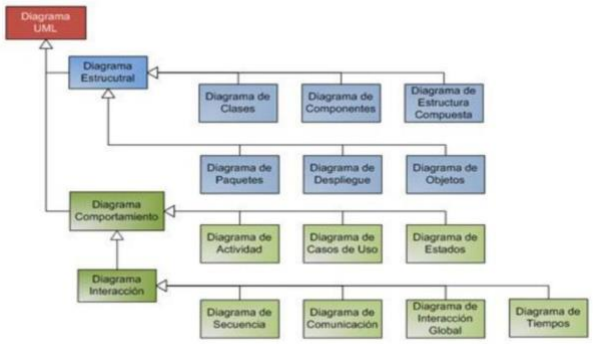
Por que precisamos de UML?

- Eles fornecem uma representação visual do problema ou de todo o sistema. Desta forma, mesmo um não desenvolvedor pode entender como o sistema funciona.
- Os diagramas UML também desempenham um papel fundamental na solução de problemas. Podemos representar um problema do mundo real e gradualmente encontrar uma maneira de resolvê-lo.
- Como estes diagramas são simples de entender, não é necessário ter conhecimento prévio para saber como uma entidade funciona ou como um processo flui.
- Utilizando um diagrama UML holístico, toda a equipe de desenvolvimento de software pode colaborar e trabalhar em conjunto.
- Eles têm muitas aplicações em programação orientada a objetos, desenvolvimento web, prototipagem, análise de negócios, entre outras.

Tipos de diagramas UML

UML não é definido em um único tipo de diagrama ou princípio. Ao invés disso, é a abordagem unificada para fornecer diferentes tipos de representações visuais. Existem diferentes tipos de diagramas UML com os quais podemos trabalhar, dependendo de nossas necessidades.

Os tipos de diagramas UML podem ser distinguidos como estruturais ou comportamentais. Cada uma dessas classificações pode ter seus próprios tipos.



A seguir, uma breve descrição dos 14 tipos de diagramas cobertos pela UML.

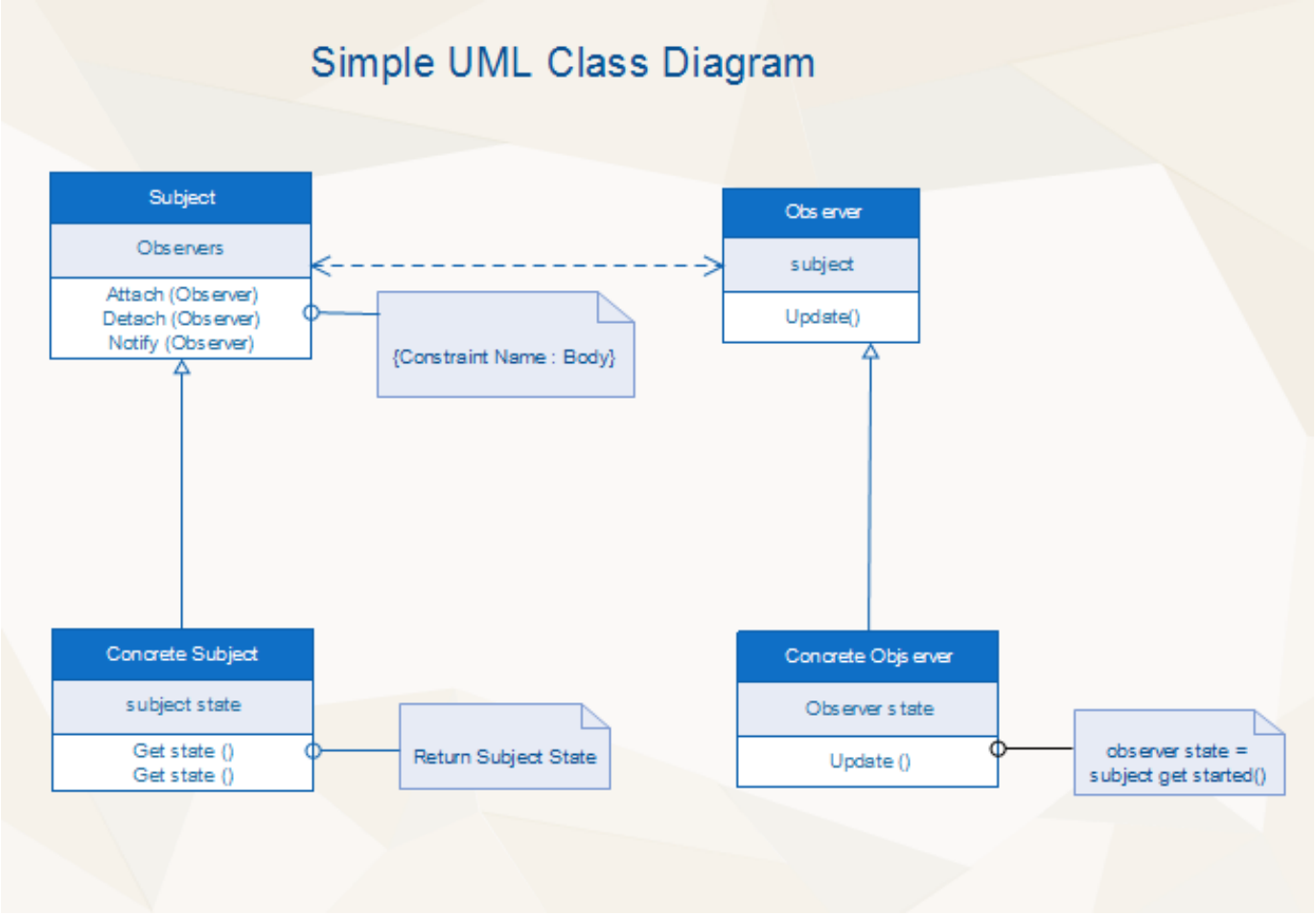
Diagramas estructurais UML

Como o nome indica, elas representam a estrutura geral de um sistema: seus objetos, atributos e relações entre as diferentes entidades. Ou seja, estamos mais preocupados com o que é o sistema do que com a forma como ele se comporta.

- **Diagrama de classes**: Descreve os diferentes tipos de objetos em um sistema e as relações entre eles. Dentro das classes mostra as propriedades e operações, assim como as restrições nas conexões entre objetos.
- **Diagrama de objetos**: Fotos dos objetos de um sistema em um momento no tempo.
- **Diagrama de pacotes**: Mostra a estrutura e as dependências entre pacotes, que permitem agrupar elementos (não apenas classes) para a descrição de grandes sistemas.
- **Diagrama de implantação**: Mostra a relação entre os componentes ou subsistemas de software e o hardware onde ele é implantado ou instalado.
- **Diagrama de estrutura composta**: Hierarquicamente decompõe uma classe mostrando sua estrutura interna.
- **Diagrama de componentes**: Ele mostra a hierarquia e as relações entre os componentes de um sistema de software.

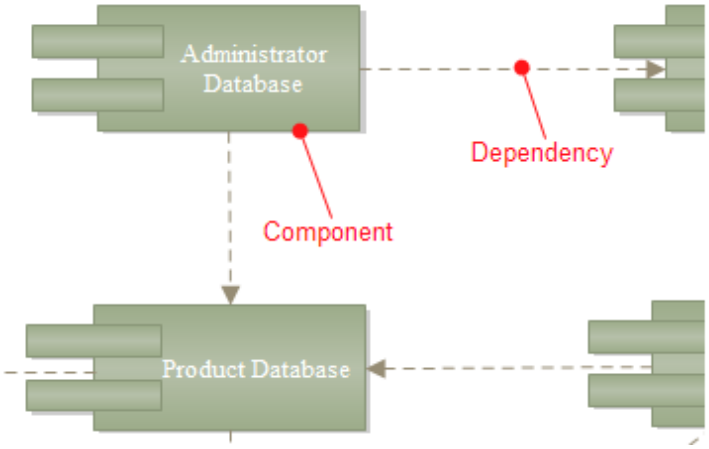
1. Diagrama de classes

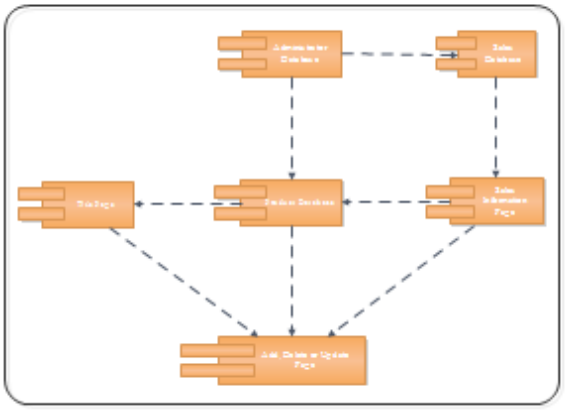
Este diagrama UML básico representa a bifurcação de um sistema em classes individuais. Utilizamos estes tipos de diagramas UML para fornecer a representação estática do programa. Uma classe tem três elementos principais: seu nome, seus atributos e seu comportamento.



2. Diagrama de componentes

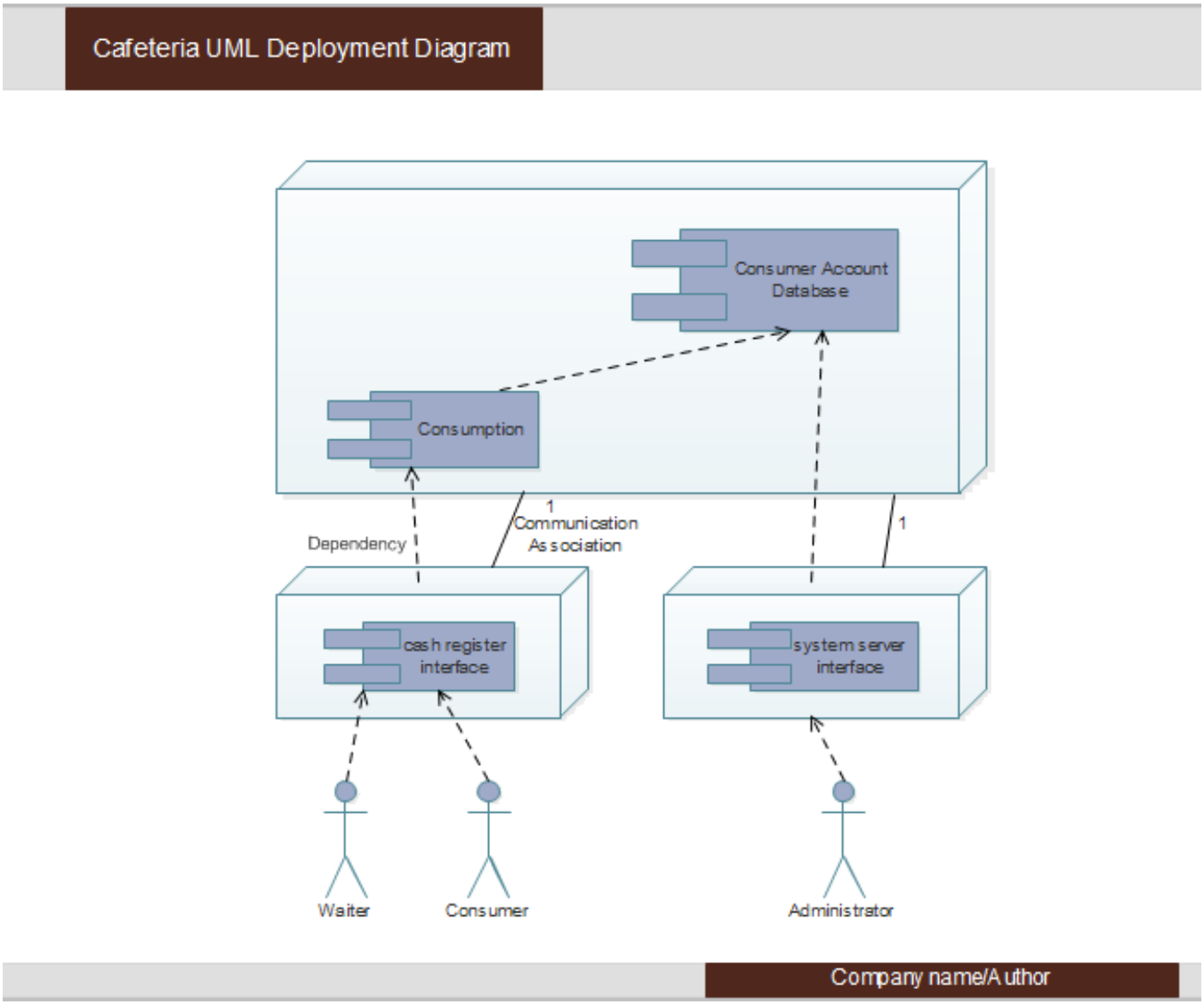
Os diagramas de componentes são criados principalmente quando um sistema é complexo e consiste em demasiadas classes. Portanto, dividimos o sistema inteiro em diferentes componentes e mostramos como eles dependem uns dos outros. Estes tipos de diagramas UML consistem de apenas duas coisas, os componentes e as dependências.





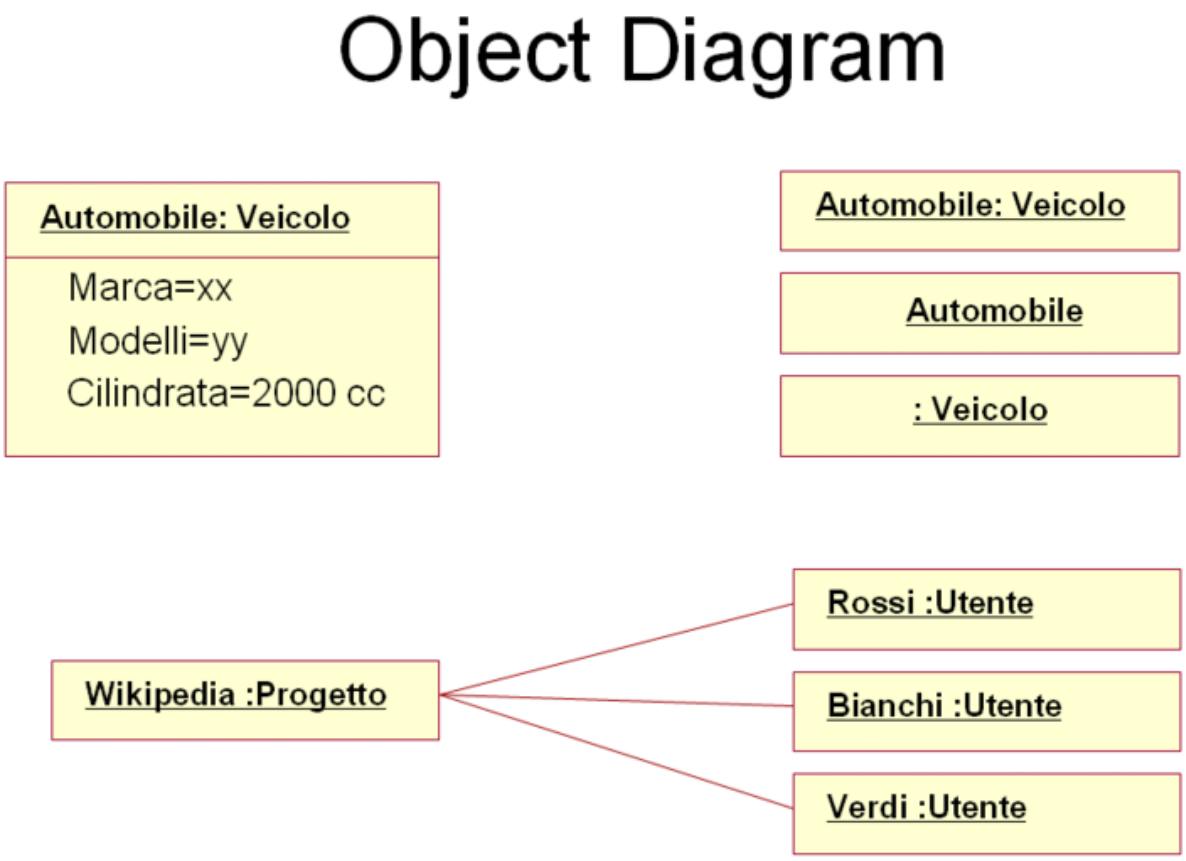
3. Diagrama de implantação

Isto fornece a representação real do hardware do sistema (com a inclusão do componente de software no mesmo). Este diagrama UML segue uma abordagem mais realista de como o sistema seria implantado. Ela consiste em duas unidades principais: nós e artefatos. Os nós representam o servidor de banco de dados ou a unidade de hardware, enquanto os artefatos representam os clientes ou esquemas.



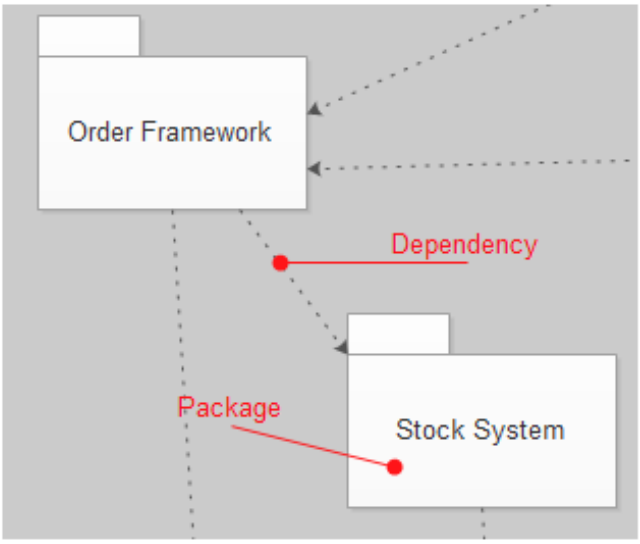
4. Diagrama de objetos

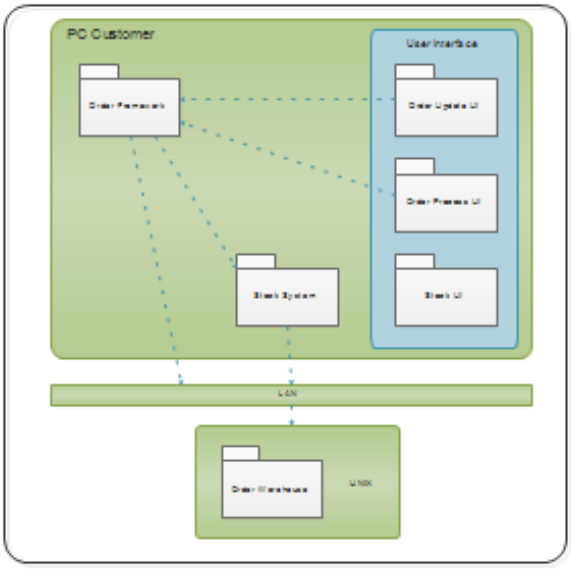
Considere isto como uma quebra de um diagrama de classes (já que uma classe é um conjunto de objetos diferentes). Entretanto, esses tipos de diagramas UML são baseados principalmente em entidades do mundo real. Ele apresenta diferentes objetos com seus elos. Cada entidade tem um nome de objeto e seu conjunto de atributos.



5. Diagrama de paquetes

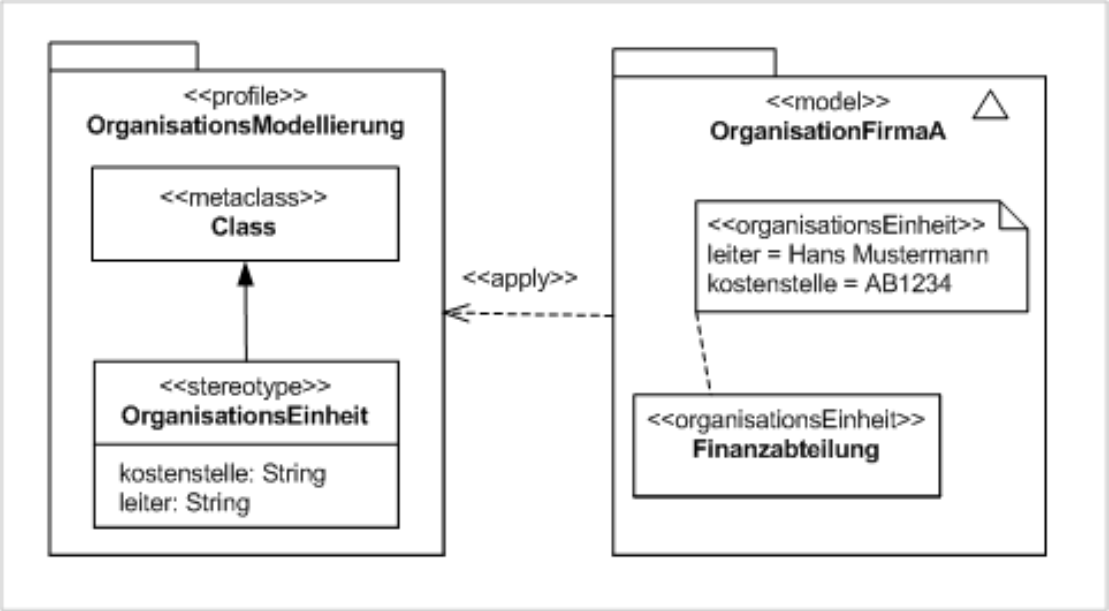
Isto fornece uma representação de um sistema com seus módulos e subsistemas de nível superior. Em um diagrama de pacotes, podemos incluir várias estruturas de diagramas de implantação. Todo o sistema pode ser facilmente decomposto e seus componentes podem ser ligados. Um pacote é representado por um ícone de pasta com um nome. Um pacote pode ter diferentes pacotes para representar uma abordagem de cima para baixo.





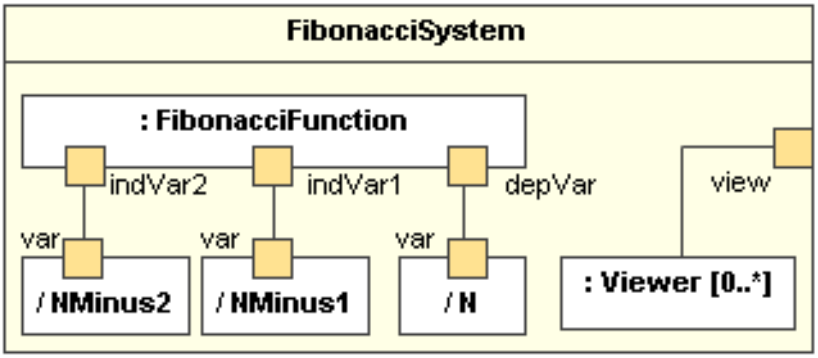
6. Diagrama de perfil

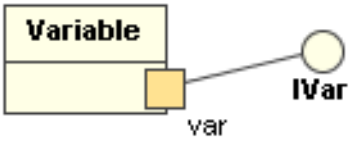
Este é um novo tipo de diagrama UML que foi introduzido no UML 2. Embora estes tipos de diagramas UML não sejam tão populares, eles podem ser usados para representar a meta estrutura do sistema. Ou seja, quais são os principais perfis, meta classes e assim por diante. Pode haver conexões dentro dos elementos de um perfil e entre diferentes perfis.



7. Diagrama da estrutura composta

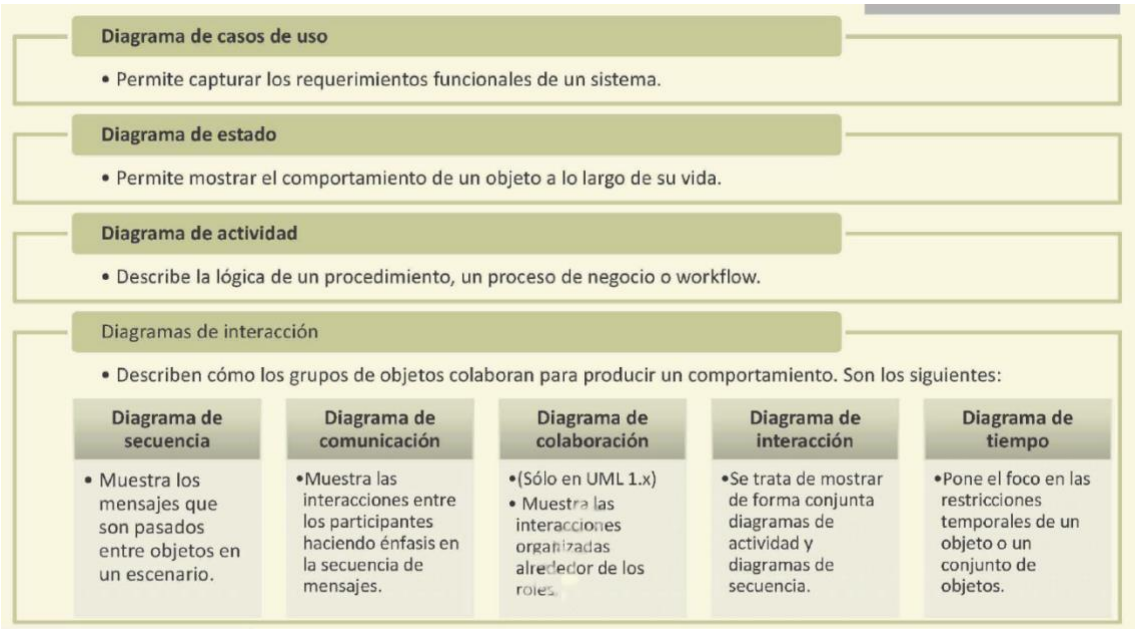
Estes tipos de diagramas UML são usados para representar a estrutura interna de uma classe. Com eles podemos ver como as diferentes entidades de uma classe se relacionam entre si e como a própria classe está associada com a entidade ou sistema externo. Cada elemento tem um "rol" atribuído aqui.





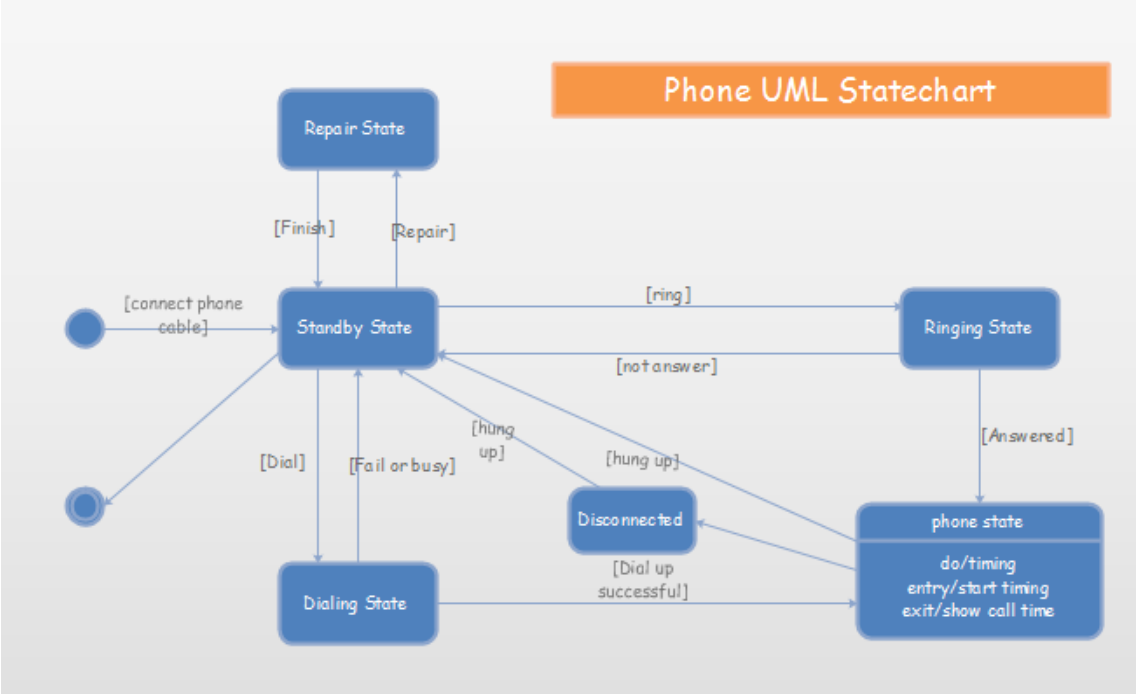
Diagramas UML de comportamento

Estes diagramas definem o comportamento do sistema e a funcionalidade geral de cada unidade. Além disso, alguns destes diagramas também representam o fluxo de informações e controle no sistema.



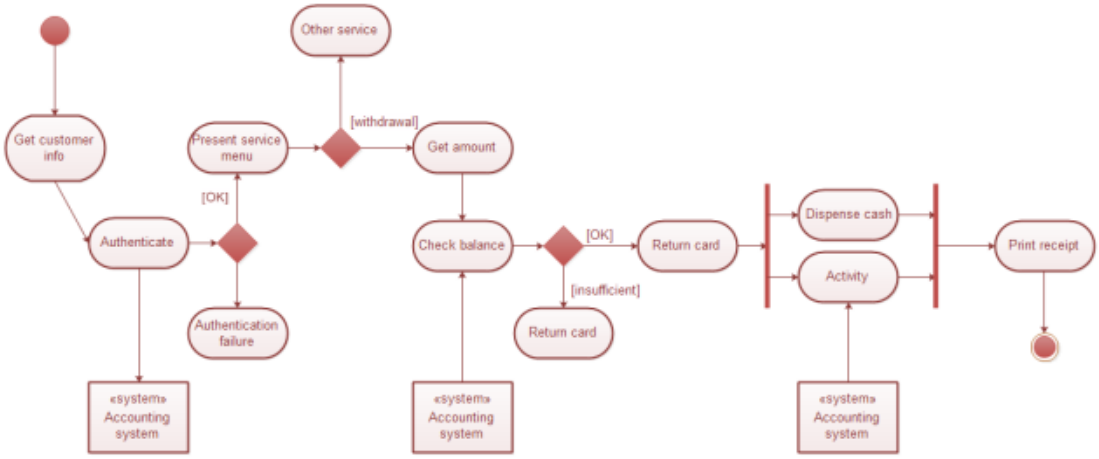
8. Diagrama de máquina de estado

Este é um diagrama dinâmico que descreve os diferentes estados do sistema. Ela apresenta todos os tipos de ações e como uma classe responderia a elas. Conseqüentemente, é estabelecido um fluxo focalizado no estado das entidades centrais.



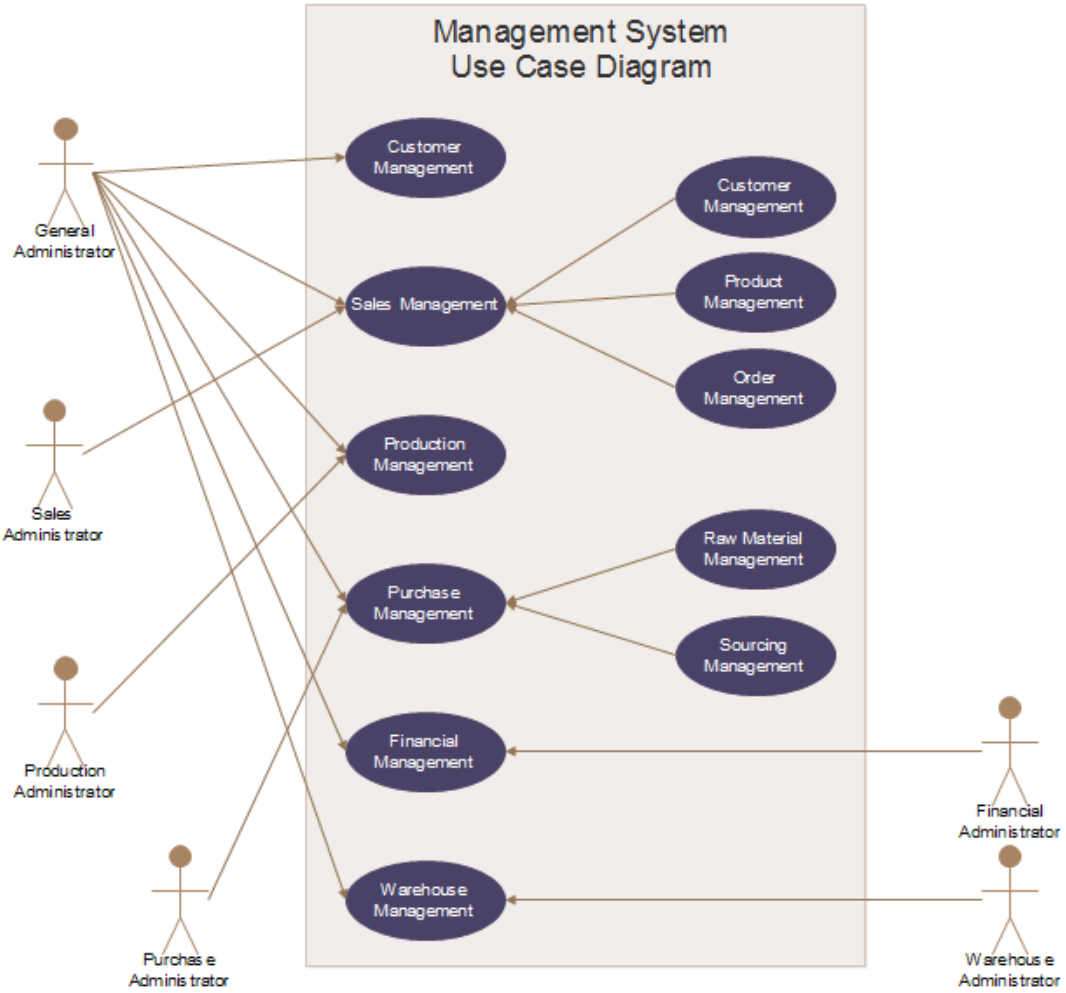
9. Diagrama de atividade

Esses tipos de diagramas UML são amplamente utilizados, pois representam o fluxo geral do sistema em relação a numerosas atividades. Ele toma um exemplo do sistema e registra como ele reage. Estes diagramas se concentram mais na implementação do sistema no mundo real do que em sua estrutura.



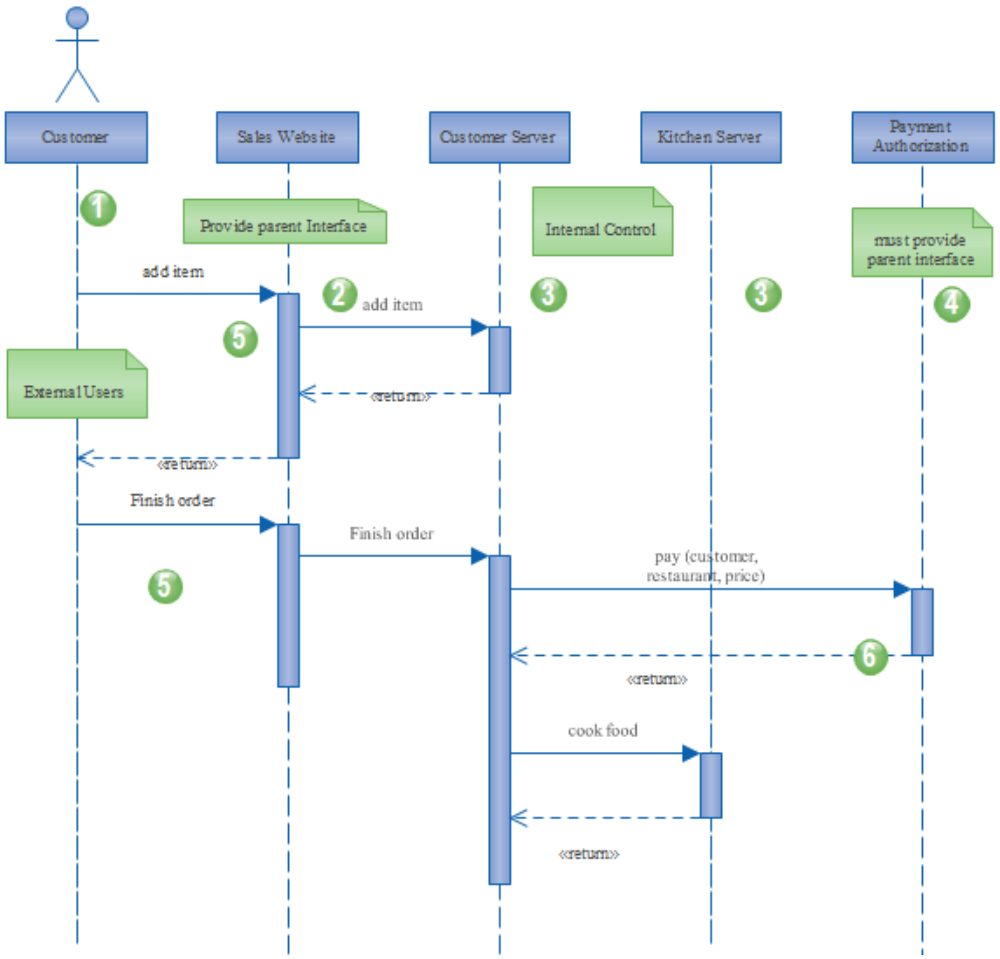
10. Diagrama de casos de uso

Este é um dos tipos mais populares de diagramas UML representando como o usuário final interagiria com o sistema. Ele apresenta diferentes atores-chave (usuários) e sua interação com componentes específicos do sistema. Além do desenvolvimento de software, ele também desempenha um papel fundamental em sua implantação.



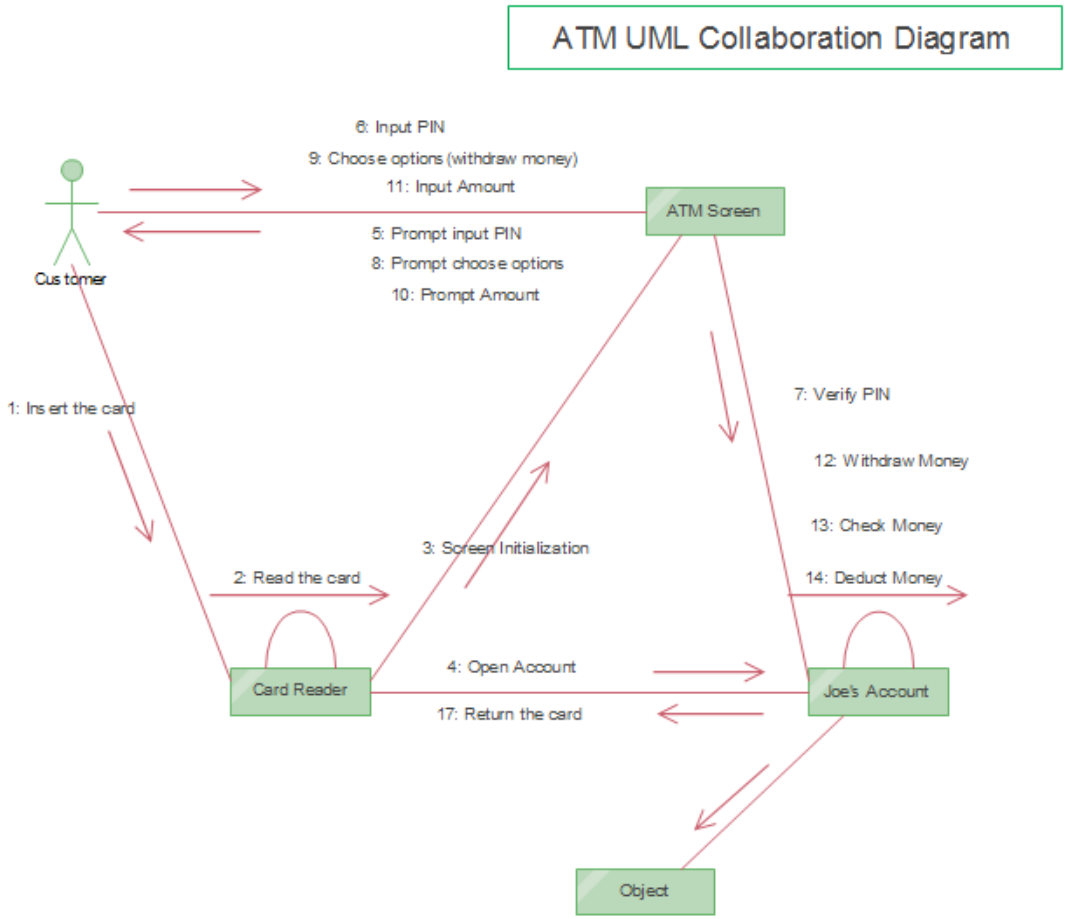
11. Diagrama seqüencial

Como o nome sugere, ele descreve o fluxo de um processo entre vários componentes de forma sequencial. É usado principalmente no desenvolvimento da arquitetura e na implementação do sistema no mundo real. Ele apresenta diferentes componentes e descreve como o fluxo passa de um componente para outro.



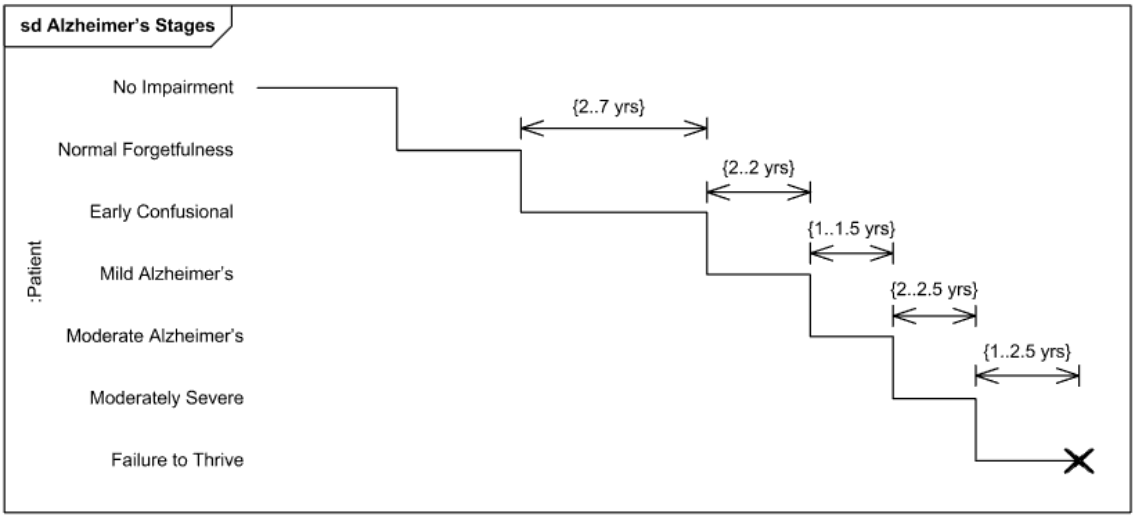
12. Diagrama de comunicação

Também conhecido como diagrama de colaboração, ele representa de maneira simples como diferentes entidades em um sistema se comunicam umas com as outras. Além de objetos, também apresenta setas e pontos numéricos para mostrar a comunicação geral no sistema.



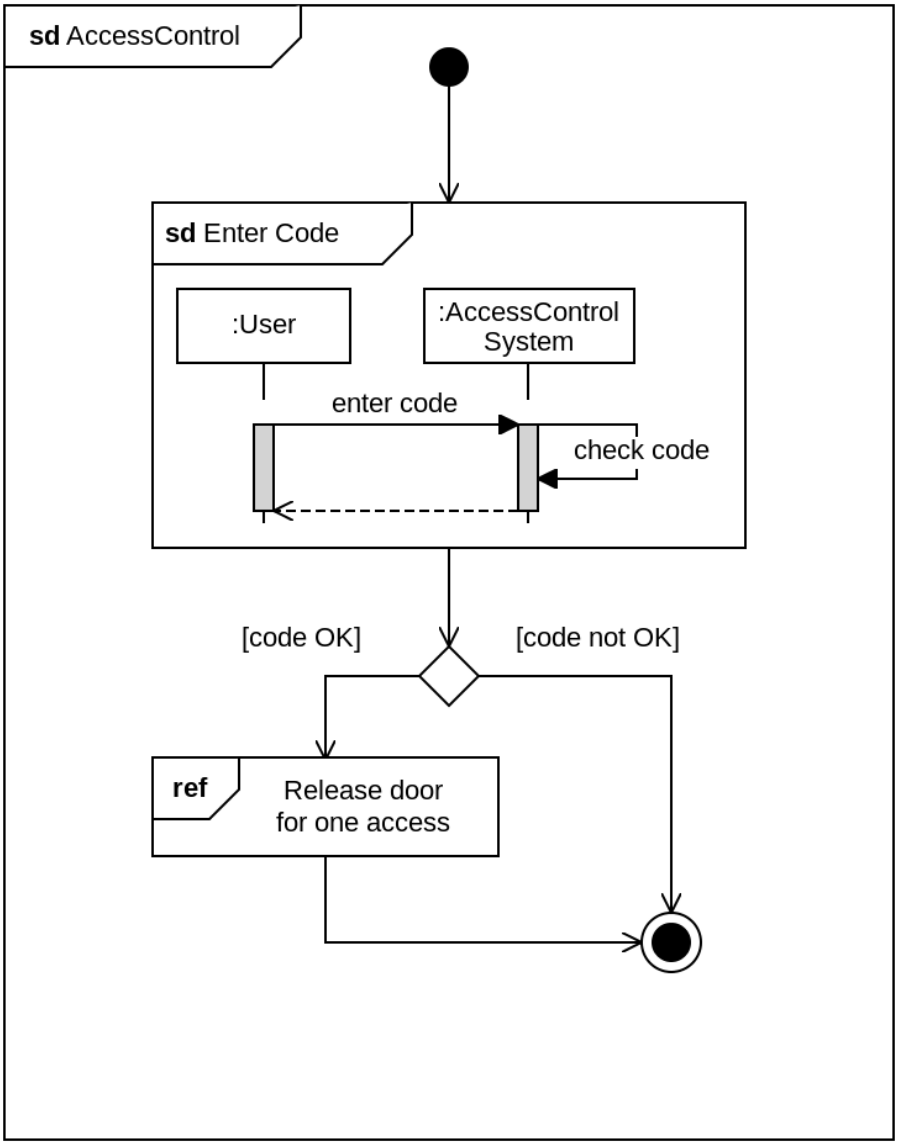
13. Diagrama de tempos

Embora semelhantes aos diagramas de seqüência, eles representam o comportamento de qualquer objeto durante uma duração específica. Estes tipos de diagramas UML foram adicionados ao UML 2 e se concentram no tempo como uma restrição.



13. Diagrama geral interativo

A princípio, esses tipos de diagramas UML podem parecer semelhantes aos diagramas de atividade. Entretanto, em vez de atividades, elas mostram o fluxo de várias seqüências interativas. Uma atividade é representada por uma estrutura que pode ter diferentes unidades dentro dela.



Diagramas de classes

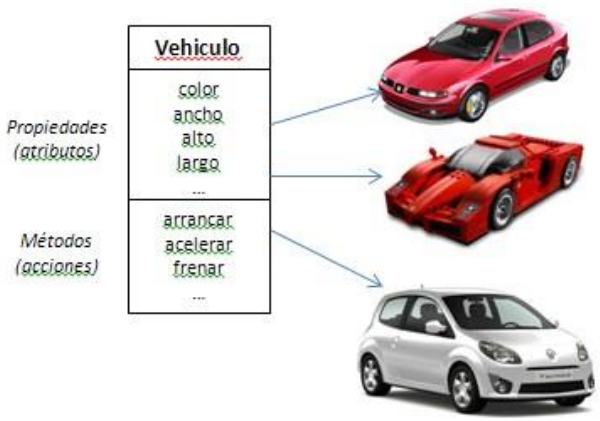
Uma vez que tenhamos visto as características gerais dos diferentes tipos de diagramas UML, vamos dar uma olhada mais profunda nos diagramas de classes, já que eles são os mais utilizados para representar o código escrito com OOP e o tipo de diagramas que usaremos principalmente como programadores. A propósito, aproveitaremos a oportunidade para rever as principais características dos elementos da linguagem Python.

O diagrama de classes mostra os componentes básicos de qualquer sistema orientado a objetos. Os diagramas de classe representam uma visão estática do modelo, ou parte do modelo, descrevendo quais atributos e comportamento ele tem, em vez de detalhar os métodos para alcançar as operações. Os diagramas de classes são muito úteis para ilustrar as relações entre as classes e as interfaces. Generalizações, agregações e associações são valiosas para refletir a herança, composição ou uso e conexões, respectivamente.

Os diagramas de classes mostram as diferentes classes que compõem um sistema e como elas se relacionam umas com as outras. Diz-se que os diagramas de classe são "estáticos" porque mostram as classes, juntamente com seus métodos e atributos, assim como as relações estáticas entre elas: quais classes "conhecem" outras classes, ou quais classes "fazem parte" de outra classe, embora não mostrem as chamadas de método entre elas.

Uma classe define os atributos e métodos de um conjunto de objetos. Todos os objetos de uma classe (instâncias da classe) compartilham o mesmo comportamento e têm o mesmo conjunto de atributos (cada objeto tem seu próprio conjunto). s vezes é usado o termo "tipo" em vez de "classe", mas é importante ter em mente que eles não são iguais, pois "tipo" é um termo mais geral.

Em UML, as classes são representadas por retângulos com o nome da classe, que podem mostrar os atributos e operações da classe em dois "compartimentos" dentro do retângulo.



Classes:

- + attr1: int
- +attr2: stirng
- + operation1(p : bool) : double
- # operation2 ()

Representação visual de uma classe em UML.

Atributos

Em UML, os atributos são exibidos pelo menos com seu nome, mas também podem exibir seu tipo, valor inicial e outras propriedades. Os atributos também podem ser exibidos com sua visibilidade:

+ representa atributos **públicos**

representa atributos **protegidos**

- representa atributos **privados**

Métodos

As operações (métodos) também são mostradas pelo menos com seu nome, embora também possam mostrar seus parâmetros e os tipos que retornam. As operações podem, como atributos, ser mostradas com sua visibilidade:

+ representa operaciones **públicas**

representa operaciones **protegidas**

- representa operaciones **privadas**

Associações de classes

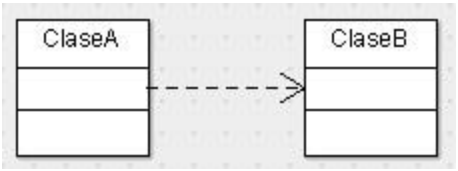
As classes podem se relacionar (estar associadas) umas com as outras de maneiras diferentes:

Dependência

(...depende de ...)(... restrito a...) (...usa...)

É uma relação de uso entre duas classes (uma usa a outra). Esta relação é a mais básica entre as classes e comparada com os outros tipos de relações, a mais fraca.

Ela é representada por uma seta tracejada que começa de uma classe e aponta para outra. A direção da seta indica quem usa quem.



Pelo diagrama acima, podemos ver que:

- A ClaseA utiliza a ClaseB.
- A ClaseA depende da ClaseB.
- Dada a dependência, qualquer mudança na ClaseB pode afetar a ClaseA.
- A ClaseA sabe que a ClaseB existe, mas a ClaseB não sabe que a ClaseA existe.

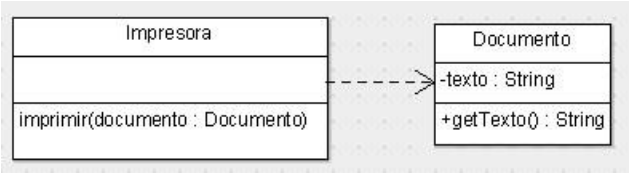
Exemplo prático:

Temos uma classe de Impresora.

Temos uma classe Documento com um atributo de texto.

A classe da impressora é responsável pela impressão dos Documentos.

Para isso, geramos uma relação de dependência:

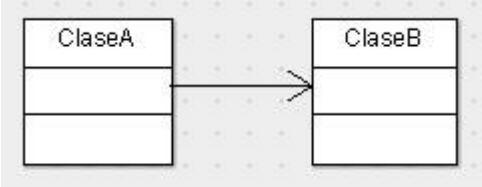


Associação

(...conoce...)(...trabalha para...)

É uma relação estrutural entre classes, ou seja, uma entidade é construída a partir de uma ou mais outras entidades. Embora este tipo de relação seja mais forte que a Dependência, é mais fraca que a Agregação, já que a vida útil de um objeto não depende de outro.

É representada por uma seta contínua que começa de uma classe e aponta para outra. A direção da flecha indica a classe que é composta (base da flecha) e seus componentes (ponta da flecha).



Pelo diagrama acima, podemos ver que:

- A ClaseA depende da ClaseB.
- A ClaseA está associada à ClaseB.
- A ClaseA sabe que a ClaseB existe, mas a ClaseB não sabe que a ClaseA existe.
- Qualquer mudança na ClaseB pode afetar a ClaseA.

Isto significa que a ClaseA deve ter como atributo um objeto ou instância do ClaseB (seu componente). A ClaseA pode acessar as funcionalidades ou atributos de seu componente, utilizando seus métodos.

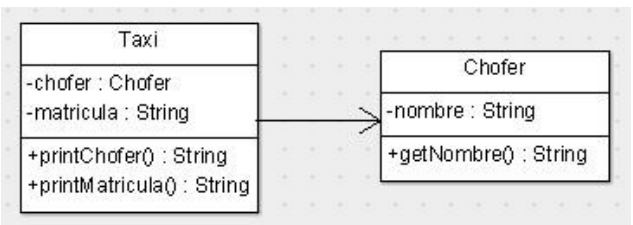
Exemplo prático:

Temos uma classe Taxi com um atributo de placa de carro.

Temos uma classe Chofer de classe com um nome de atributo.

Cada Taxi precisa ser conduzido por um Chofer.

Taxi precisa ter acesso a alguns dos atributos de seu Chofer (por exemplo, seu nome).

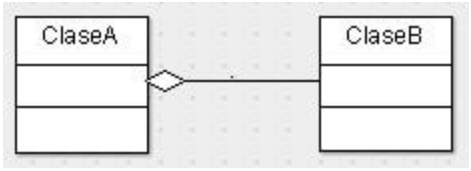


Agregação

(...é um acessório de...)

É muito semelhante à relação da Associação, só que varia na multiplicidade, pois em vez de ser uma relação "um a um", é uma relação "um a muitos".

É representado por uma seta de uma classe para outra com um losango branco na base.



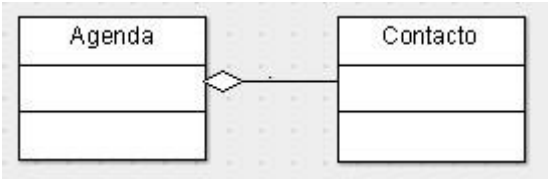
O ClaseA agrupa vários elementos do tipo ClaseB.

Exemplo:

Temos uma classe Agenda.

Temos uma classe Contacto.

Uma Agenda reúne vários Contactos.

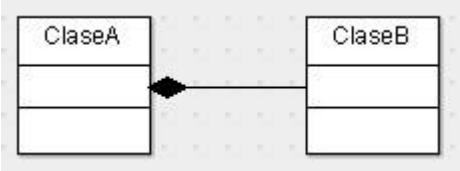


Composição

(...faz parte de...)

Semelhante à relação de Agregação apenas que a Composição é uma relação mais forte. Ela fornece documentação conceitual por ser uma "relação vitalícia", ou seja, a vida útil de um objeto é condicionada pela vida útil do objeto que o inclui.

É representado por uma seta de uma classe para outra com um losango preto em sua base.



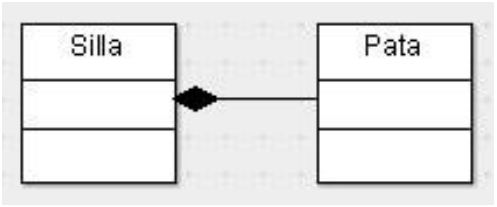
- O ClaseA agrupa vários elementos do tipo ClaseB.
- A vida útil dos objetos do tipo ClaseB é condicionado pela vida útil do objeto do tipo ClaseA.

Exemplo:

Temos uma classe Silla.

Um objeto Silla é, por sua vez, composto de quatro objetos do tipo Pata.

O tempo de vida útil dos objetos Pata depende da vida útil do Silla, porque se não há Silla não pode haver um Patas.



Generalização

(...es um...)

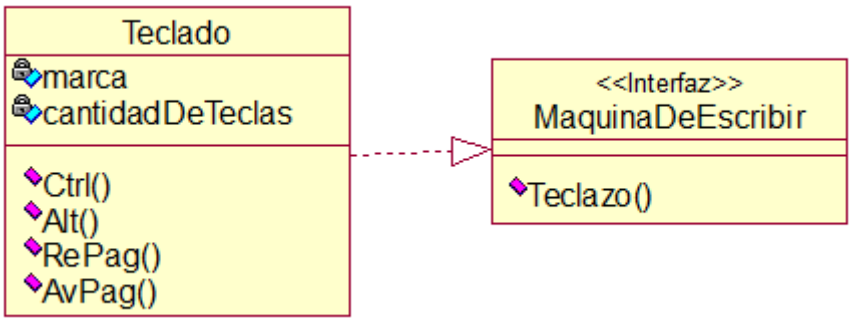
É uma relação de herança. Pode-se dizer que é um "é um tipo de" relação.

Em UML, a herança é representada por uma seta, cujo ponto é um triângulo vazio. A seta que representa a herança é orientada de subclasse para superclasse.

Realização: (Interfaces)

Uma classe especifica um conjunto de condições que a outra classe tem que cumprir. Esta é a representação típica das interfaces. Uma interface é um conjunto de operações que especifica um determinado aspecto da funcionalidade de uma classe, e é um conjunto de operações que uma classe apresenta a outras classes. E a relação entre uma classe e uma interface é conhecida como uma realização.

A realização é indicada por uma linha tracejada com uma seta sem preenchimento no final, ou também pode ser representada por um pequeno círculo que se conecta com uma linha a uma classe.



Exemplo de diagrama UML: