



# Iniciando com UML



que é a Linguagem UML?

**UML**, abreviação de Unified Modeling Language, é uma linguagem de modelagem padronizada que consiste em um conjunto integrado de diagramas, desenvolvido para ajudar os desenvolvedores de sistemas e software a especificar, visualizar, construir e documentar os artefatos de sistemas de software, bem como modelagem de negócios e outros. A UML representa uma coleção das melhores práticas de engenharia de software que se mostraram bem-sucedidas na modelagem de sistemas grandes e complexos. A UML é uma parte muito importante do desenvolvimento de software orientado a objetos e do processo de desenvolvimento de software. A UML usa principalmente notações gráficas para expressar o design de projetos de software. O uso da UML ajuda as equipes de projeto a se comunicar, explorar possíveis projetos e validar o design arquitetônico do software.

## Quais são os diagramas de UML ?

Existem três classificações de diagramas UML:

A figura 1 apresenta todos os diagramas:

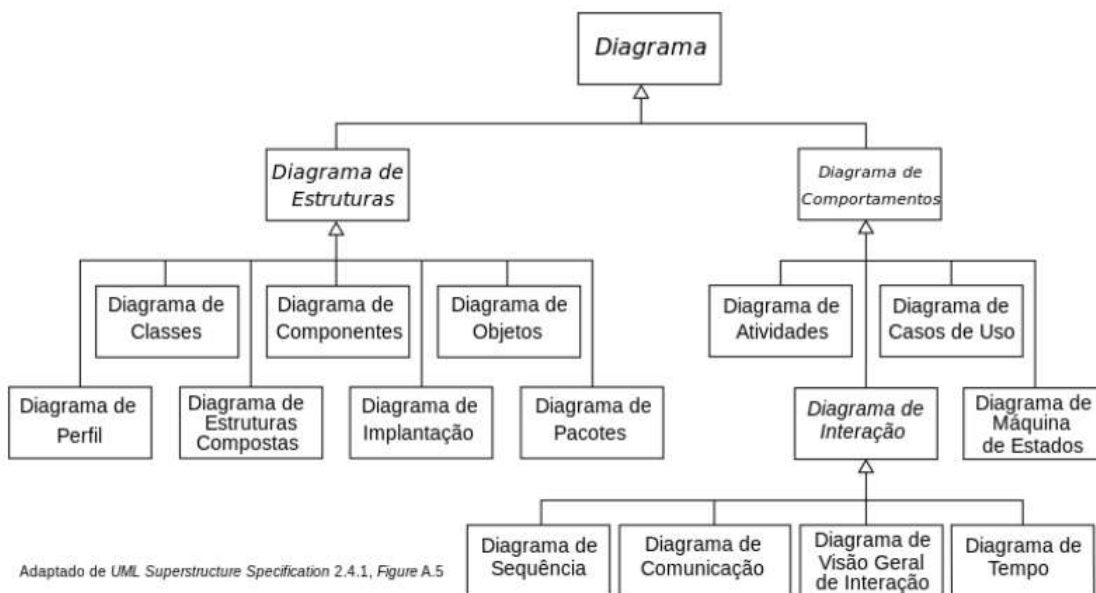
A figura 1 apresenta todos os diagramas:

- **Diagramas de comportamento:** Um tipo de diagrama que descreve recursos comportamentais de um sistema ou processo de negócios. Isso inclui atividade,

máquina de estado e diagramas de casos de uso, bem como os quatro diagramas de interação.



- **Diagramas de interação:** Um subconjunto de diagramas de comportamento que enfatizam as interações com objetos. Isso inclui comunicação, visão geral da interação, sequência e diagramas de tempo.
- **Diagramas de estrutura:** Um tipo de diagrama que descreve os elementos de uma especificação independentemente do tempo. Isso inclui diagramas de classe, estrutura composta, componente, implantação, objeto e pacote.



**Figura 1 – Divisão dos diagramas de UML**

**Fonte: Autoral**


A tabela 1 resume os treze diagramas e a coluna de prioridade indica a importância do aprendizado.

Diagrama	Descrição	Prioridade de Aprendizagem 
Diagrama de atividades	Descreve processos de negócios de alto nível, incluindo fluxo de dados	Alto
Diagrama de Classes	Mostra uma coleção de elementos de modelo estático, como classes e tipos, seu conteúdo e seus relacionamentos.	Alto
Diagrama de Comunicação	Mostra instâncias de classes, suas inter-relações e o fluxo de mensagens entre elas. Os diagramas de comunicação geralmente se concentram na organização estrutural dos objetos que enviam e recebem mensagens. Anteriormente chamado de diagrama de colaboração.	Baixo
Diagrama de componentes	Descreve os componentes que compõem um aplicativo, sistema ou empresa. Os componentes, suas inter-relações, interações e suas interfaces públicas são representados.	Médio
Diagrama de estrutura composta	Descreve a estrutura interna de um classificador (como uma classe, componente ou caso de uso), incluindo os pontos de interação do classificador com outras partes do sistema.	Baixo
Diagrama de implantação	Mostra a arquitetura de execução dos sistemas. Isso inclui nós, ambientes de execução de hardware ou software, bem como o middleware que os conecta.	Baixo
Diagrama de visão geral da interação	Uma variante de um diagrama de atividades que mostra o fluxo de controle em um sistema ou processo de negócios. Cada nó / atividade dentro do diagrama pode representar outro diagrama de interação.	Baixo
Diagrama de Objetos	Descreve objetos e seus relacionamentos em um determinado momento, normalmente um caso especial de um diagrama de classes ou de comunicação.	Baixo
Diagrama de Pacotes	Mostra como os elementos do modelo são organizados em pacotes, bem como as dependências entre os pacotes.	Médio
Diagrama de sequência	Modela a lógica sequencial, com efeito a ordem do tempo das mensagens entre os classificadores.	Alto
Diagrama da Máquina de Estado	Descreve os estados em que um objeto ou interação pode estar, bem como as transições entre estados. Anteriormente chamado de diagrama de estados, diagrama de estados ou diagrama de transição de estados.	Médio
Diagrama de tempo	Descreve a alteração no estado ou condição de uma instância ou função do classificador ao longo do tempo. Normalmente usado para mostrar a alteração no estado de um objeto ao longo do tempo em resposta a eventos externos.	Baixo
Diagrama de casos de uso	Mostra casos de uso, atores e suas inter-relações.	Alto

## Diagrama Estrutural, Diagrama de Comportamento, Diagrama de Interação

Diagramas estruturais devem ser utilizados para especificar detalhes da estrutura do sistema (parte estática), por exemplo: classes, métodos, interfaces, namespaces, serviços, como componentes devem ser instalados e como deve ser a arquitetura do sistema, entre outros

Diagramas comportamentais devem ser utilizados para especificar detalhes do comportamento do sistema (parte dinâmica), por exemplo: como as

funcionalidades devem funcionar, como um processo de negócio deve ser tratado pelo sistema, como componentes estruturais trocam mensagens e co.  respondem às chamadas, entre outros

Diagramas de interação mostram como os objetos interagem entre si. Permitem assim modelar os aspectos dinâmicos de um sistema.


## Classe e Visibilidade

Conforme mencionado na tabela anterior, o Diagrama de Classes apresenta uma coleção de elementos de modelo estático, como classes e tipos, seu conteúdo e seus relacionamentos.

**Classes:** um modelo que vai representar um conjunto de objetos que possuem comportamento e características em comum.

## Perspectivas da Classe

- Análise: em um sistema de informação para uma biblioteca, as classes podem ser, por exemplo, Livro, Usuário, Empréstimo.
- Projeto: por exemplo, classes de controle, classes de fronteiras, classes de entidades.
- Implementação: refinamentos das classes de projetos onde métodos aparecem obrigatoriamente e são considerados aspectos de controle, estereótipos, pacotes, etc.

- Visibilidade: é um recurso para dar autoridade para outros elementos dentro  orientação a objetos utilizarem, mesmo estando em outro lugar. E funciona para método, atributo, classe.

**Tipos de visibilidade:** público (public), privado (private), protegido (protected), pacote (default)

- público (public): qualquer classe do sistema pode ter acesso aos atributos e métodos definidos como public. Notação +
- privado (private): somente a própria classe pode ter acesso. Notação -
- protegido (protected): somente a própria classe e suas subclasses podem ter acesso. Notação #
- pacote (default): qualquer classe dentro do mesmo pacote pode ser acessada. Notação ~

### **Atividade extra**

Para saber mais sobre visibilidade, leia o texto: “Visibilidade em UML”, do Professor Dr. Jacques Philippe Sauvé (~jacques) da UFCG – Universidade Federal de Campina Grande. (Site da DSG/UFCG)

### **Referências Bibliográficas**

BOOCH, Grady. **Uml - Guia do Usuário**. Rio de Janeiro: Editora Campus, 2018.



GUEDES, Gilleanes T. A. **UML 2 - Uma Abordagem Prática**. São Paulo: NovaTec, 2018.

PRESSMAN, Roger; MAXIM, Bruce. **Engenharia de Software**. Porto Alegre: Bookman, 2010.

SOMMERVILLE, Ian. **Engenharia de software**. São Paulo: Pearson, 2015

**Ir para exercício**