

# Disciplina: Análise e Projeto 00

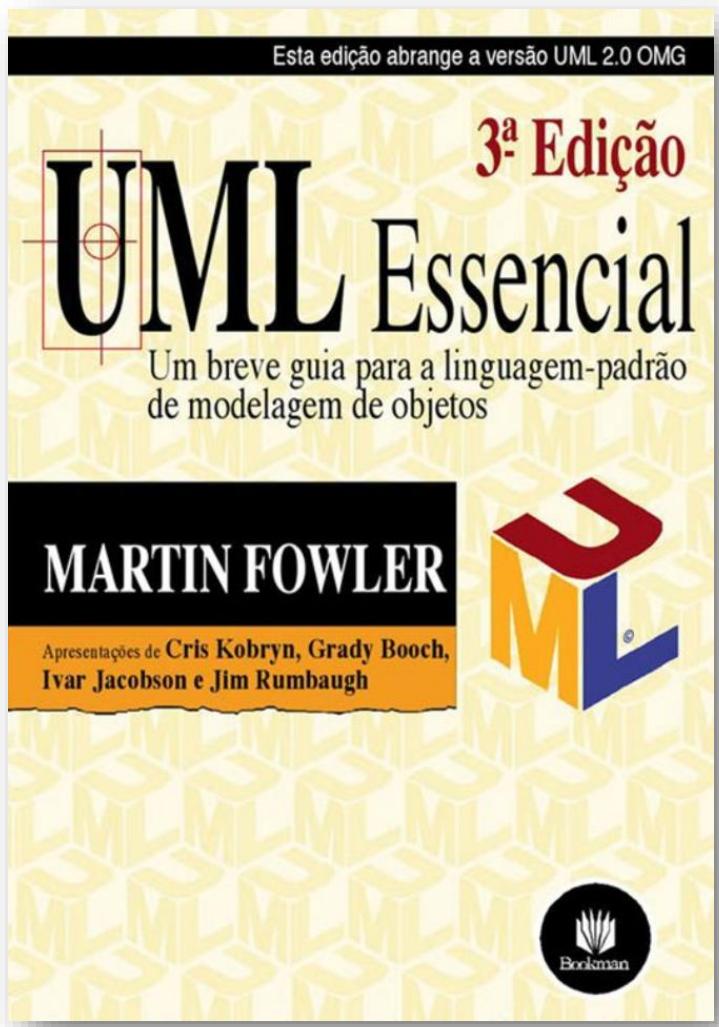
Prof. Maurício Pasetto de Freitas, MSc.



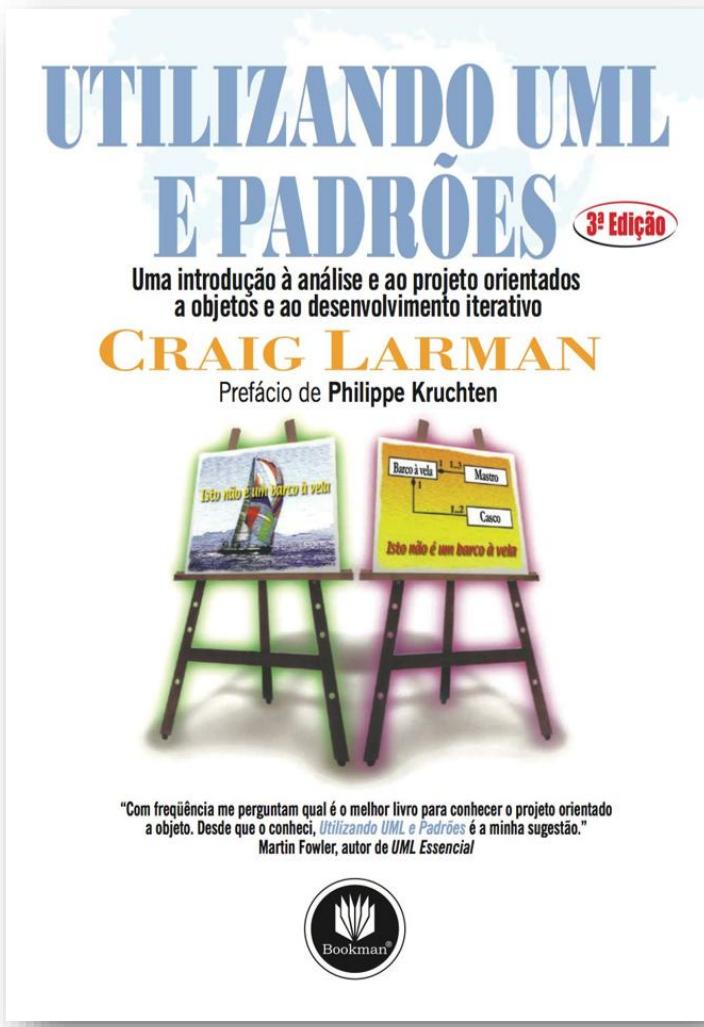


### **Formação:**

- Bacharelado Ciência da Computação – Univali – 2020;
- Especialização em Ciência de Dados e Inteligência Artificial – PUCRS – 2022;
- Mestrado em Computação Aplicada – Univali – 2023;
- Doutorando no PPEGC UFSC;
- Lattes: <https://lattes.cnpq.br/6876306606936758>



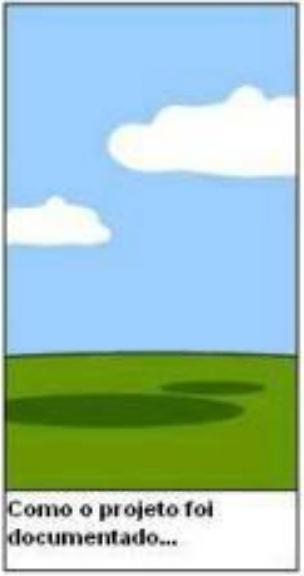
**Minha  
Biblioteca**  
.com.br



## Análise e Projeto Orientado a Objetos

Sommerville (2011), podemos chamar de “análise de sistemas” o que faz parte da **“engenharia de requisitos”**. Acrescentar o termo **“engenharia”** implica dizer que técnicas sistemáticas deverão ser utilizadas para assegurar **que os requisitos do sistema** sejam consistentes, relevantes e completos.

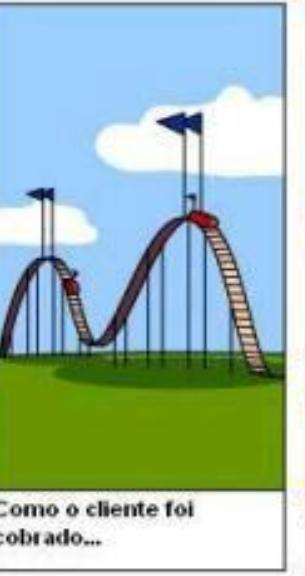
# Análise e Projeto Orientado a Objetos



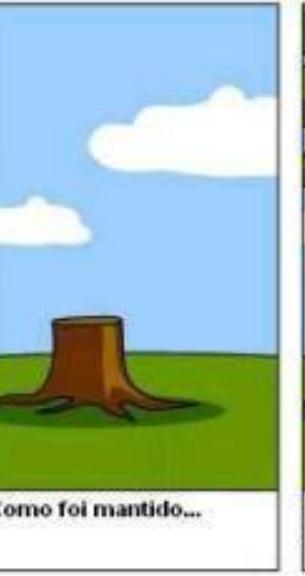
Como o projeto foi documentado...



Que funcionalidades foram instaladas...



Como o cliente foi cobrado...



Como foi mantido...



O que o cliente realmente queria...

# Análise e Projeto Orientado a Objetos



Etapas do desenvolvimento de software

## Análise e **Projeto** Orientado a Objetos

Enquanto a fase de **análise trabalha com o domínio do problema**, a fase de **projeto trabalha com o domínio da solução**, procurando estabelecer “como” o sistema fará o que foi determinado na fase de análise, ou seja, qual será a solução para o problema identificado.

### **Nesse momento serão selecionados:**

- Linguagem de programação;
- Sistema Gerenciador de Banco de Dados;
- Como será a interface do sistema;
- Software necessário para a Implantação e funcionamento correto;
- Como será a arquitetura do sistema.

## Análise e **Projeto** Orientado a Objetos

### Programação Estruturada X Orientada a Objetos

#### Programação estruturada:

- Baseada em Procedimentos;
- Controle de Fluxo;
- Divisão do Programa;
- Manutenção.

#### Exemplos de Linguagens:

- Puras: C, Pascal, Fortran, Cobol;
- Dão suporte: C++, Java, Java Script, PHP, Ruby, Python e GO.

## Onde a Programação Estruturada é utilizada nos dias de hoje?

- Sistemas embarcados e firmwares;
- Linguagens de Script e Automação;
- Desenvolvimento de Software de Baixo Nível;
- Algoritmos e Lógica de Programação;
- Projetos de Pequena Escala e Scripts Simples;
- Manutenção de Software Legado.

## Análise e **Projeto** Orientado a Objetos

### **Programação orientada a objetos:**

- Baseada em Objetos;
- Pilares:
  - Encapsulamento;
  - Abstração;
  - Polimorfismo;
  - Herança.

### **Exemplos de Linguagens:**

- C++, Java, C#, Ruby e Python.

## Programação Estruturada X Orientada a Objetos.

Diferenças:

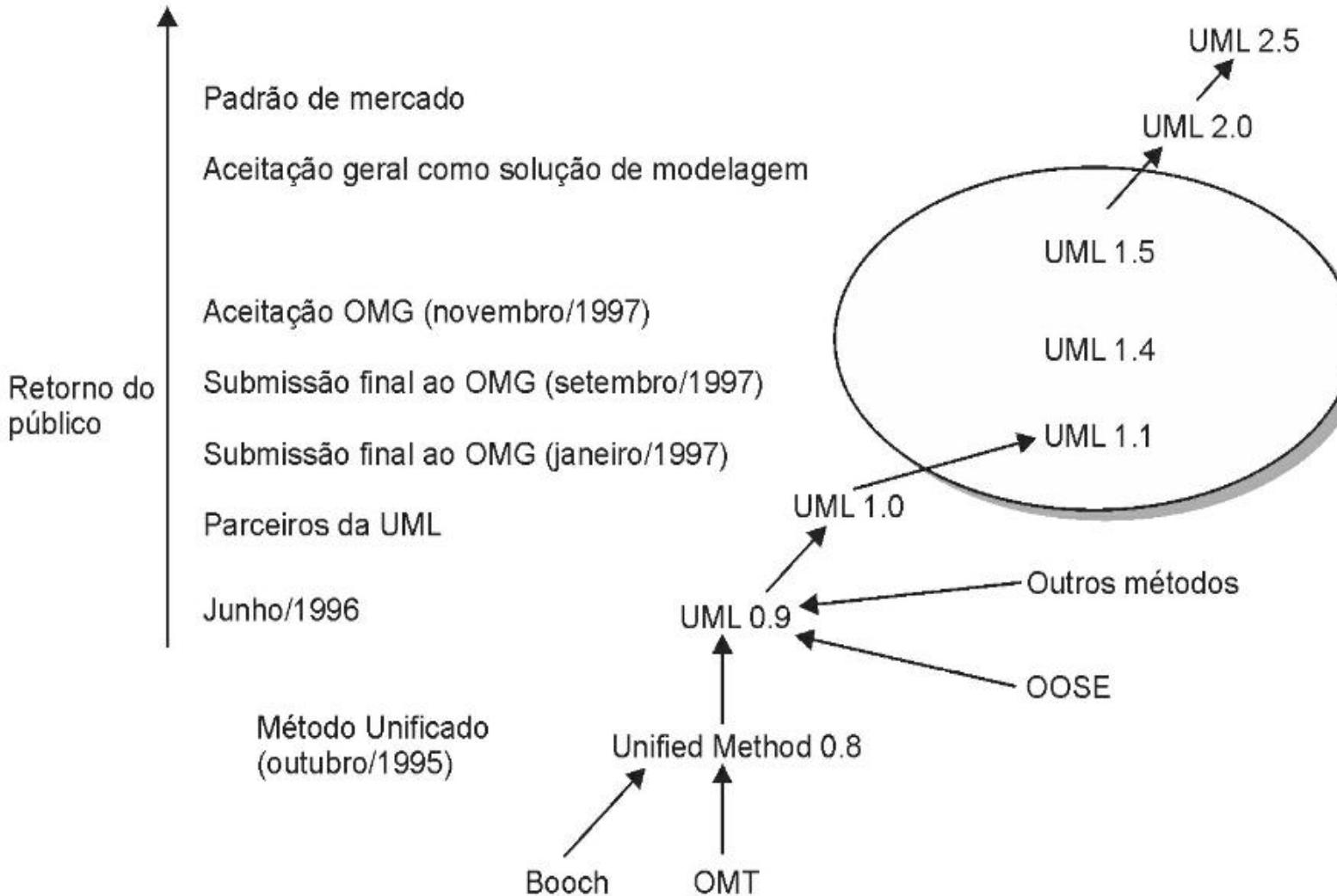
- Abordagem e Design;
- Organização do código;
- Reutilização do código;
- Manutenção e escalabilidade;
- Complexidade de aprendizado.

## **UML** – Unified Modeling Language

- É uma **linguagem visual** utilizada para **modelar softwares** baseados no paradigma de **orientação a objetos**;
- É uma linguagem de modelagem de **propósito geral** que pode ser aplicada a todos os domínios de aplicação;
- Essa linguagem é atualmente a **linguagem-padrão de modelagem adotada internacionalmente** pela indústria de engenharia de software.

# UML – Unified Modeling Language

Desde 1997, a responsabilidade pela evolução da UML ficou a cargo da **OMG** (Grupo de Gerenciamento de Objeto), órgão aprovador.



# UML – Unified Modeling Language



ABOUT US ▾ GROUPS ▾ CERTIFICATIONS ▾ RESOURCES ▾ SPECIFICATIONS ▾ COMMUNITIES ▾ MEMBERSHIP ▾

## ABOUT THE UNIFIED MODELING LANGUAGE SPECIFICATION VERSION 2.5.1

2.5.1 • UML • SPECIFICATIONS

UML®

## Unified Modeling Language

A specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems.



Specification

Title: Unified Modeling Language

Acronym: UML®

Version: 2.5.1

Document Status: formal ⓘ

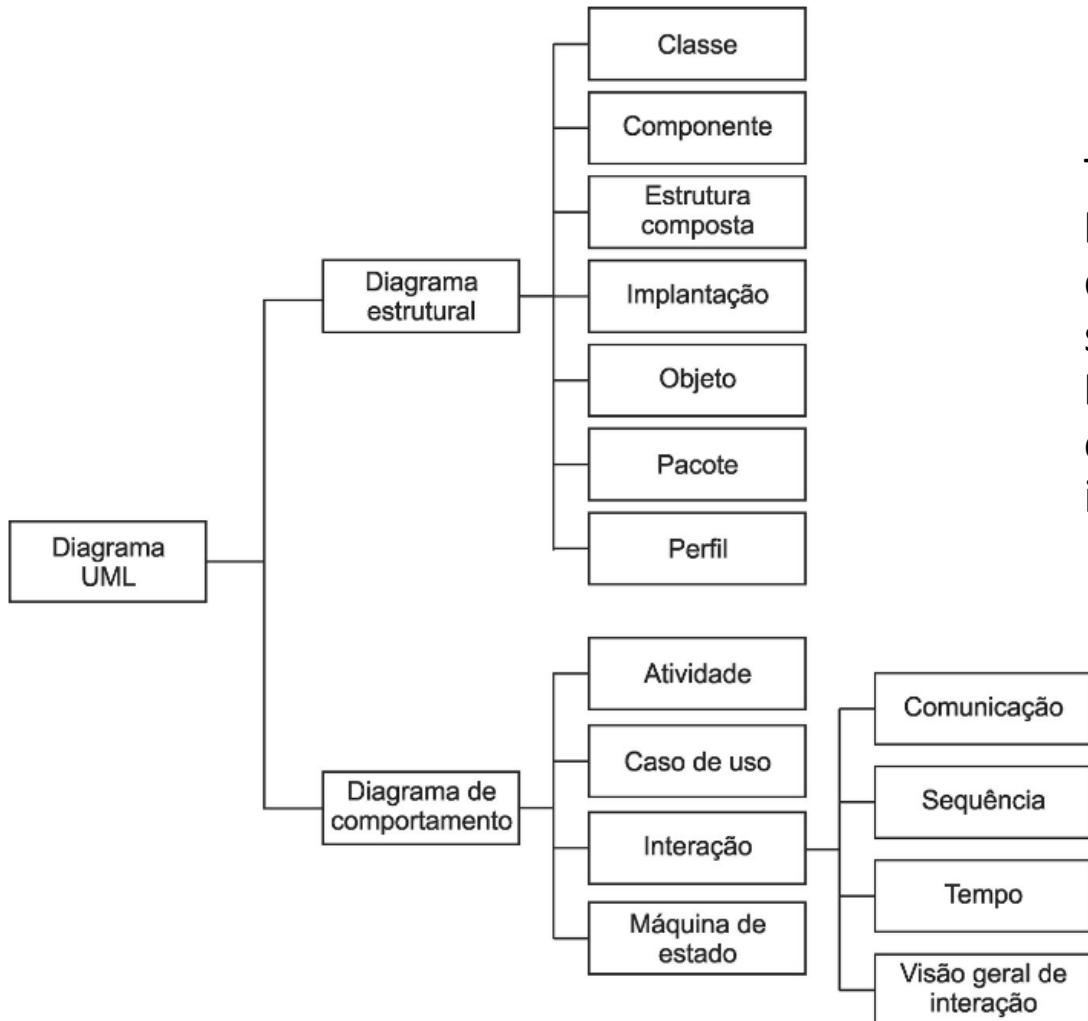
Publication Date: dezembro 2017

Categories: [Modeling](#) [Software Engineering](#) [Platform](#)

IPR Mode ⓘ RF-Limited ⓘ

<https://www.omg.org/spec/UML/>

# UML – Unified Modeling Language



Tipos de diagramas:

**Diagramas Estruturais:** descrevem os elementos estruturais que compõe o sistema;

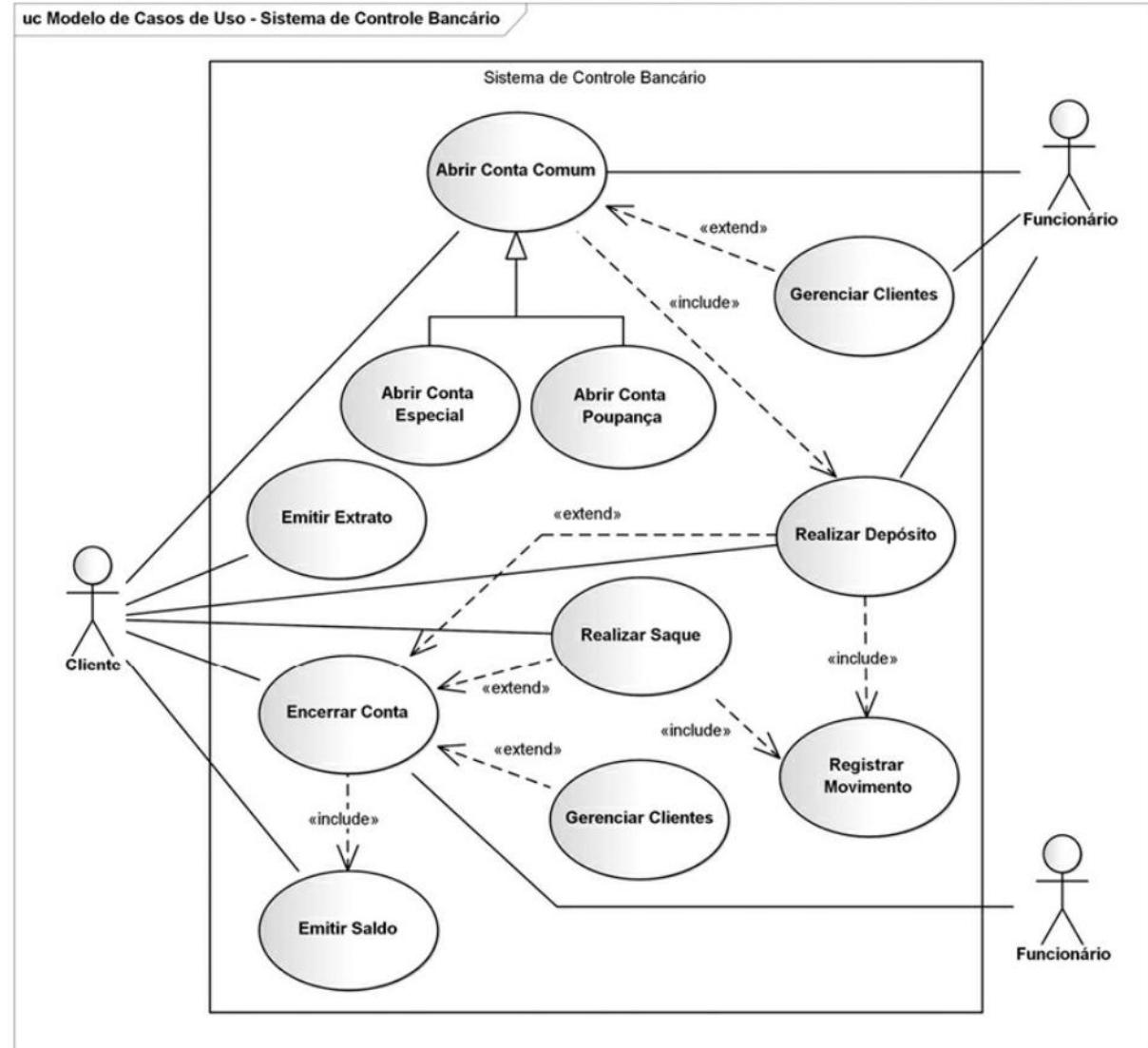
**Diagramas de comportamento:** descrevem o comportamento dos elementos e suas interações.

**Diagramas da UML**

# UML – Unified Modeling Language

## Diagrama de Casos de Uso:

- Apresenta uma visão externa geral das funcionalidades que o sistema deverá oferecer aos usuários;

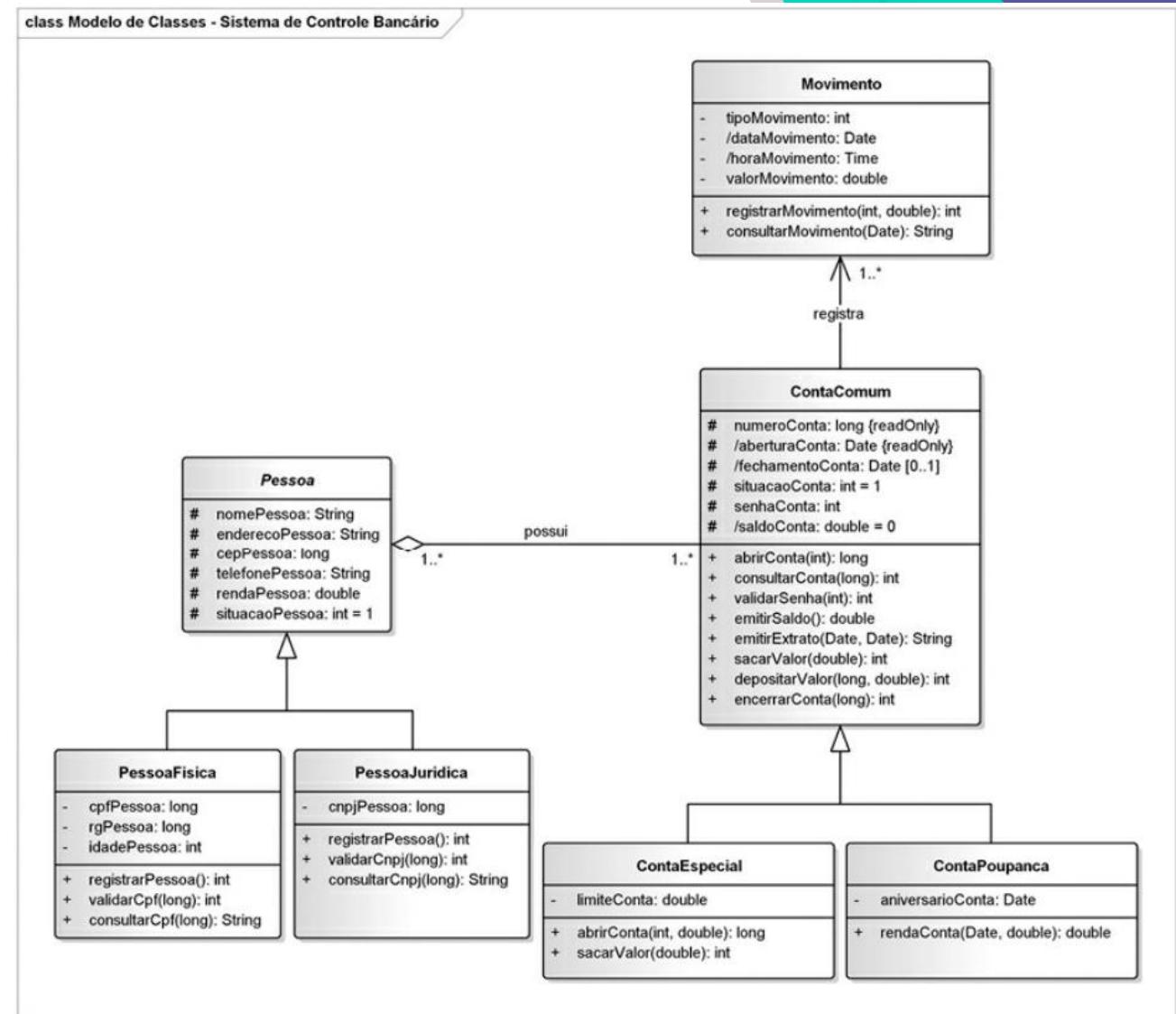


# Diagramas de Caso de Uso

# UML – Unified Modeling Language

## Diagrama de Classe:

- Permite a **visualização das classes que comporão o sistema** com seus respectivos atributos e métodos;
- Demonstrar **como as classes do diagrama se relacionam**, complementam e transmitem informações.

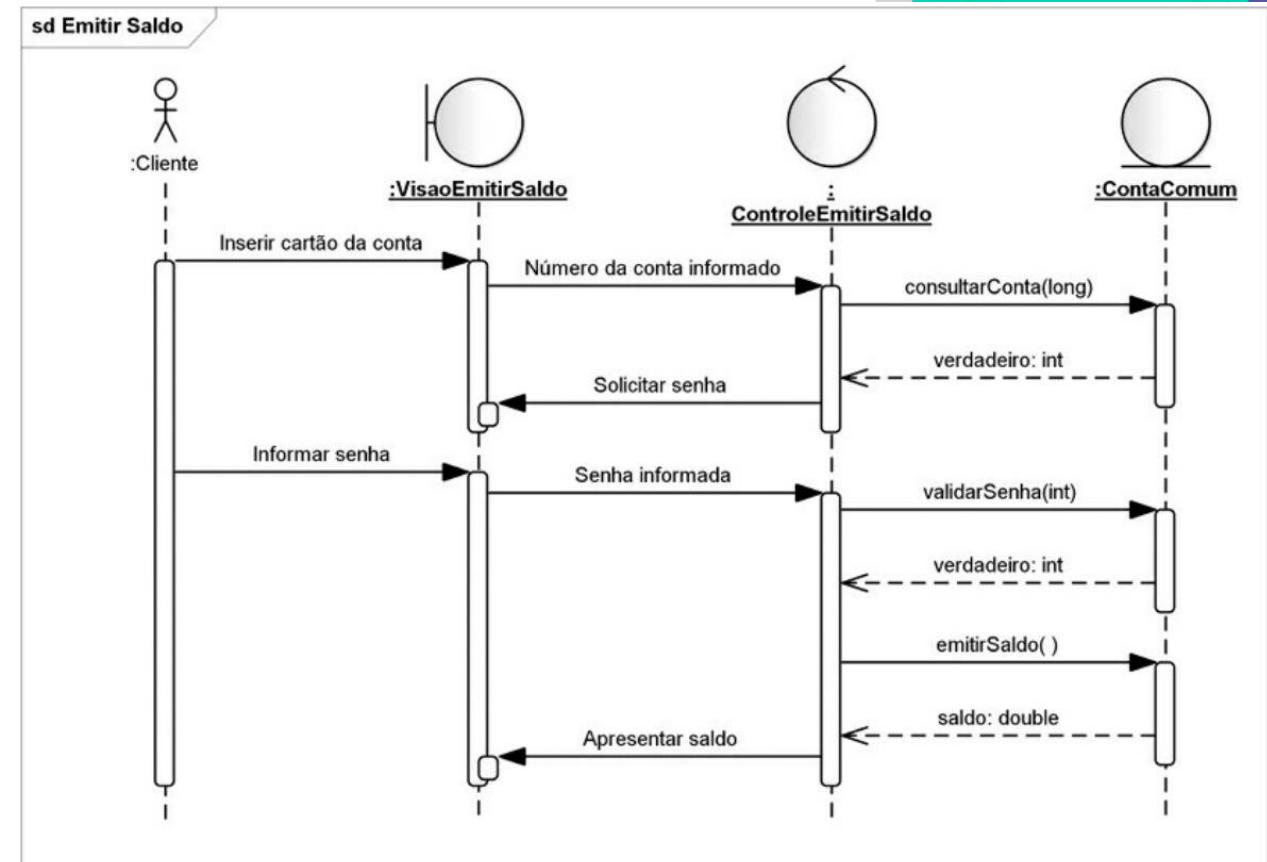


Diagramas de Classe

# UML – Unified Modeling Language

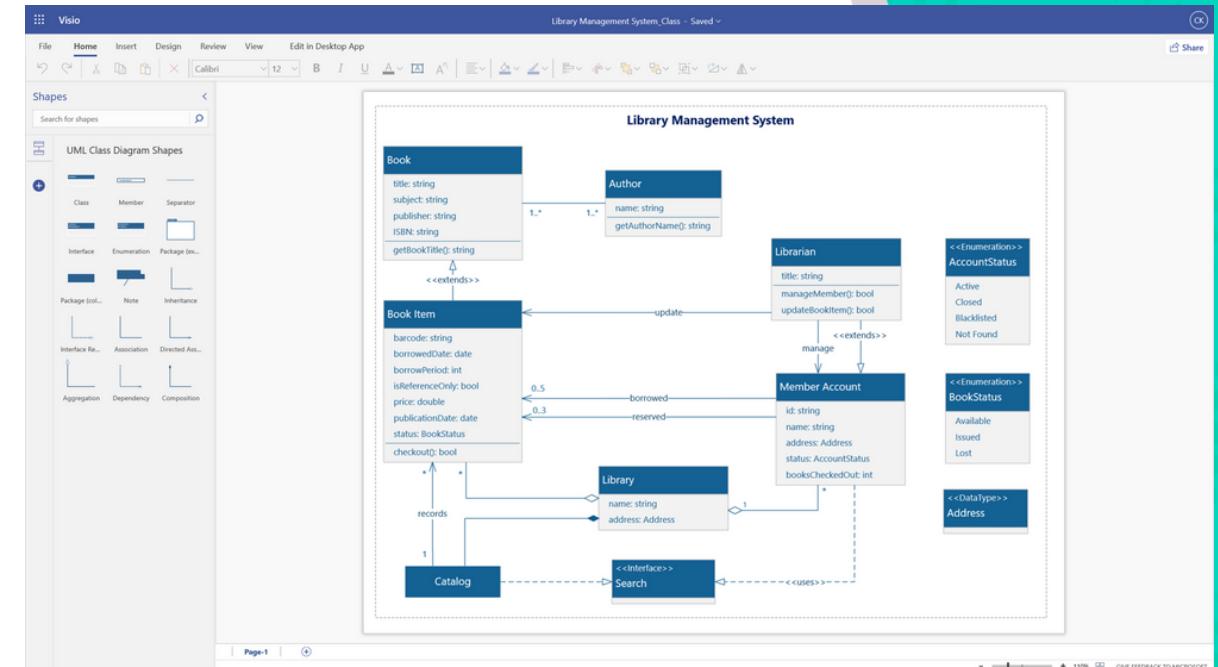
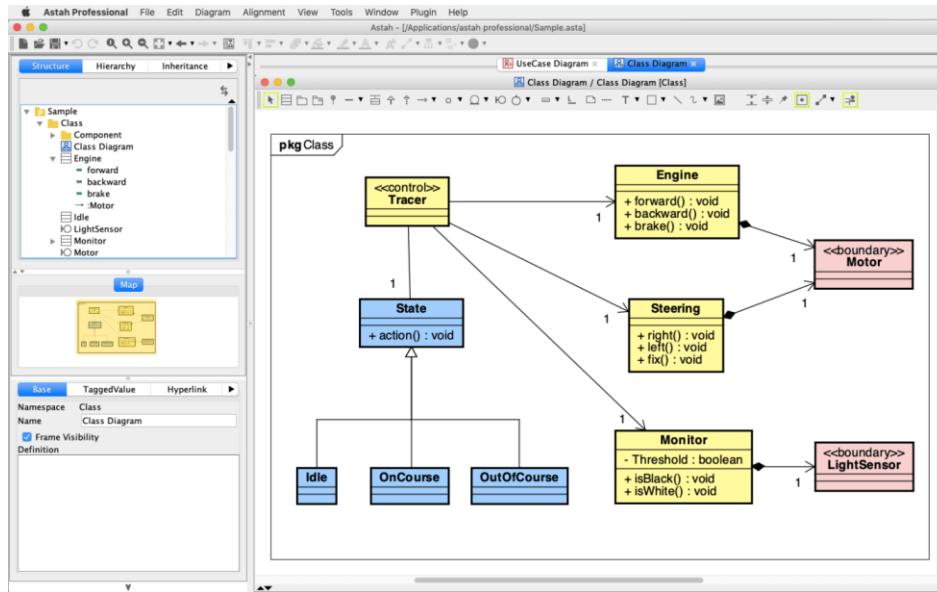
## Diagrama de Sequência:

- Procura determinar a **sequência de eventos que ocorrem** em um determinado processo, identificando quais **mensagens devem ser disparadas entre os elementos** envolvidos e em que ordem.
- **Baseia-se no diagrama de casos de uso**, havendo normalmente um diagrama de sequência para cada caso de uso declarado;



Diagramas de Sequência

# UML - Ferramentas CASE

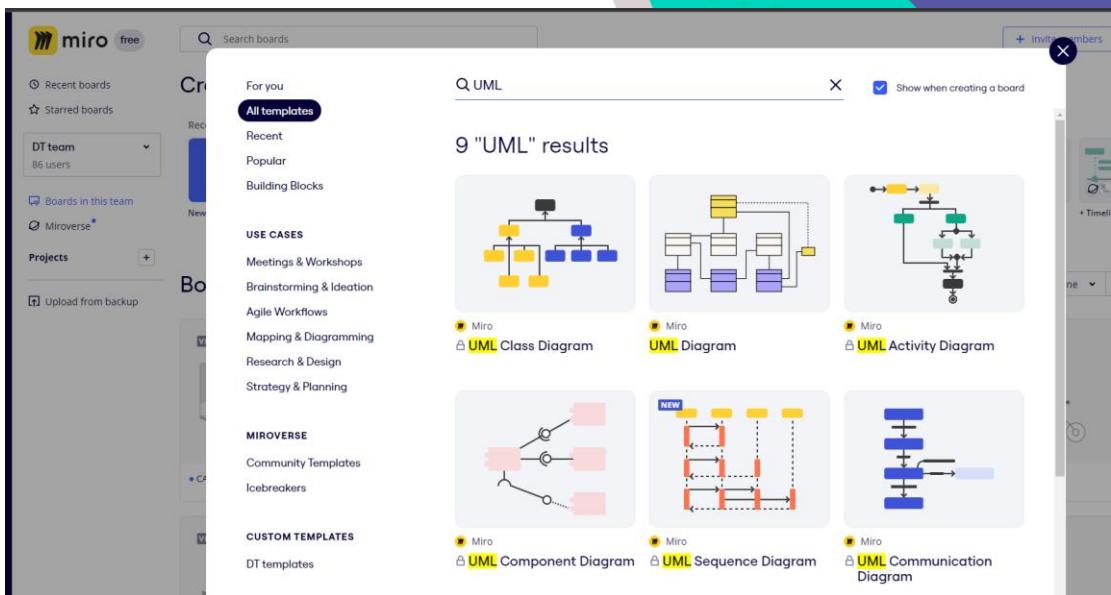


# UML - Ferramentas CASE



The diagram illustrates a UML Class Diagram with the following elements and relationships:

- Frame**: A container element at the top left.
- Classname**: A class element with three fields: `+ field1: type`, `+ field2: type`, and `+ field3: type`.
- Interface**: Three interface elements:
  - `<<Interface>> Interface` with fields `+ field1: Type` and `+ field2: Type`, and methods `+ method1(Type): Type` and `+ method2(Type, Type): Type`.
  - `<<Interface>> Interface` with fields `+ field1: Type` and `+ field2: Type`, and methods `+ method1(Type): Type` and `+ method2(Type, Type): Type`.
  - `<<Interface>> Interface` with fields `+ field1: Type` and `+ field2: Type`, and methods `+ method1(Type): Type` and `+ method2(Type, Type): Type`.
- Name**: An interface element labeled `<<interface>> Name`.
- Text**: A text element labeled `<<interface>> Name Text`.
- Relationships**: Solid arrows indicate associations between **Classname** and each of the three **Interface** elements. Dashed arrows indicate directed generalization from each of the three **Interface** elements to the **Name** and **Text** elements.



The Miro logo consists of a yellow square containing three black, slanted 'M' characters, followed by the word 'miro' in a lowercase, sans-serif font.

 **Lucidchart**

“Sucesso é o acúmulo de pequenos esforços, repetidos dia e noite.”

Robert Collier

**BONS ESTUDOS**