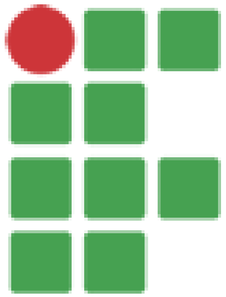


INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Disciplina

Modelagem de Domínio e Conceitos de Orientação a Objetos

PROF. FERNANDO TAMBERLINI ALVES

Revisão

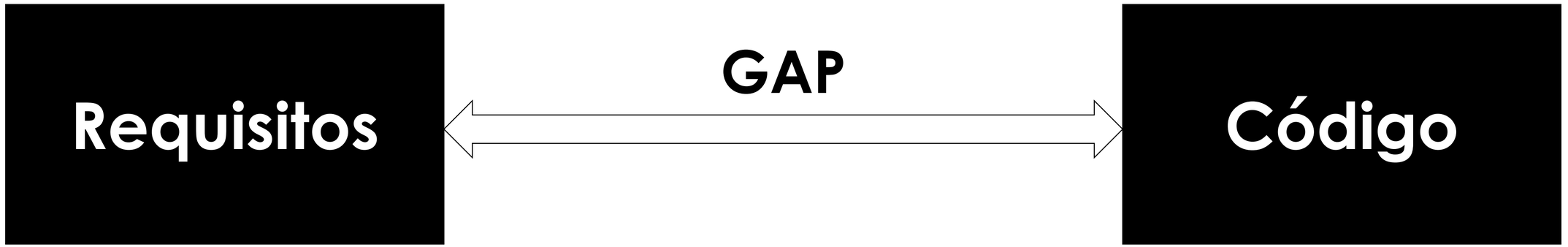
INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- **Engenharia de Software**: processos, métodos e ferramentas de apoio que nos auxiliam no desenvolvimento de software;
- **Arquitetura de Software:** características de como o software será desenvolvido;
- **Processo de Desenvolvimento:** etapas específicas a serem percorridas durante o desenvolvimento de um software (Cascata e Ágil);
- **Engenharia de Requisitos**: é o processo de entender o que o sistema deve fazer (requisitos funcionais) e como ele deve se comportar (requisitos não funcionais).
- **Modelagem: é a representação graficamente a estrutura, o comportamento e os requisitos do sistema antes de sua construção.**

Motivação

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- **Existe uma lacuna entre os requisitos e o código**



O QUE O SISTEMA FAZ
(nível de abstração mais alto)

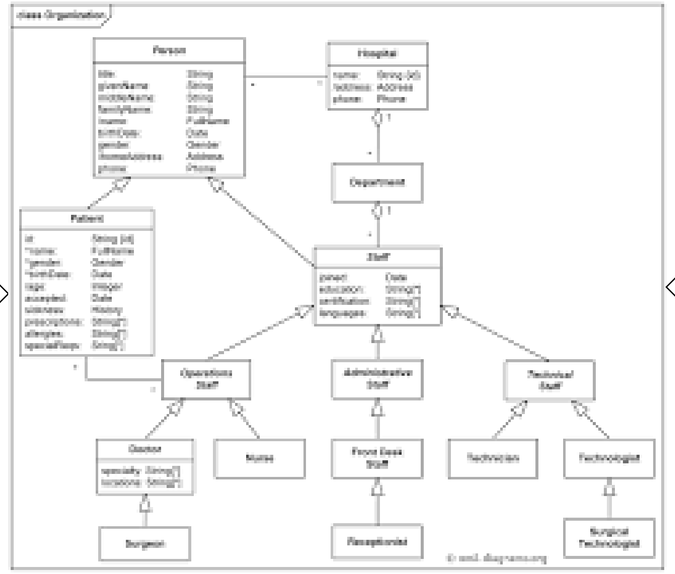
COMO O SISTEMA FAZ
(nível de abstração mais baixo)

Modelagem de Software

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Objetivo: preencher essa “lacuna”
- Documentar uma solução para o problema definido pelos requisitos

Requisitos

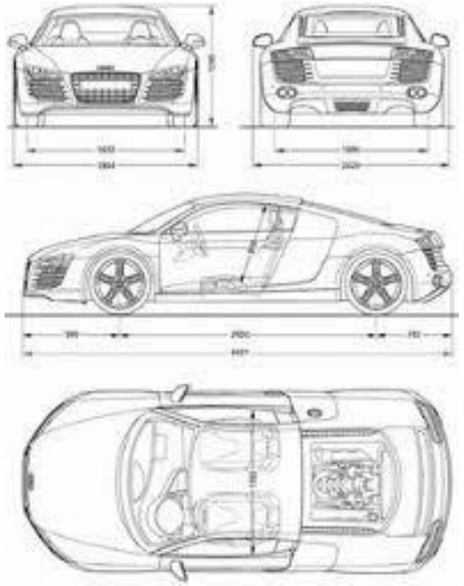


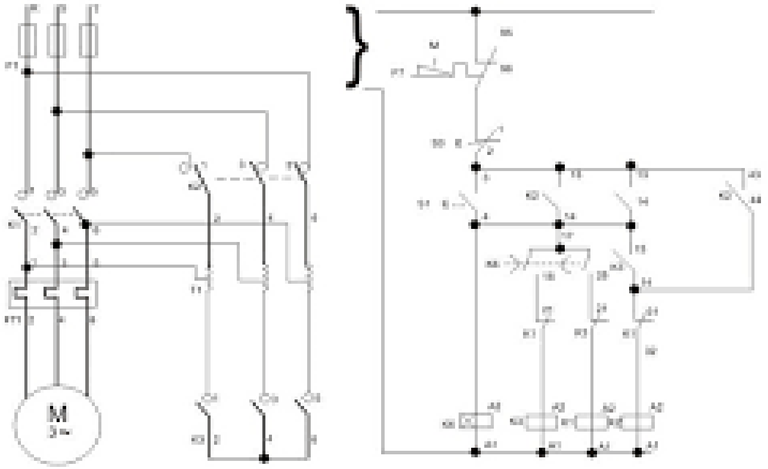
Código

O QUE O SISTEMA FAZ
(nível de abstração mais alto)

COMO O SISTEMA FAZ
(nível de abstração mais baixo)

Analogia com as outras áreas



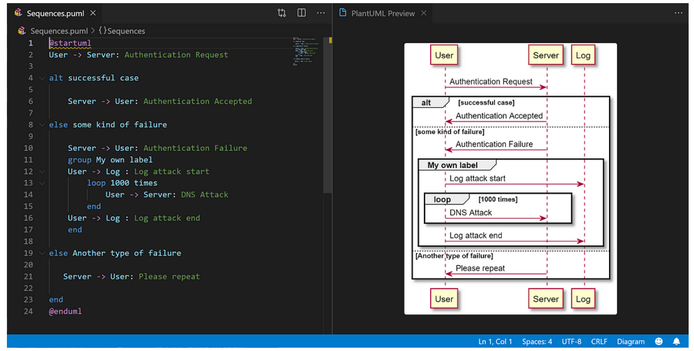


Modelos de Software

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Infelizmente, não são tão efetivos e largamente usados, como em outras engenharias

- Modelos de software podem ser:
  - Formais: menos comuns; não serão estudados aqui
  - Gráficos: UML é a notação mais comum



"muitas vezes um diagrama UML
É mais claro do que
várias linhas de código"

Modelagem com Esboço

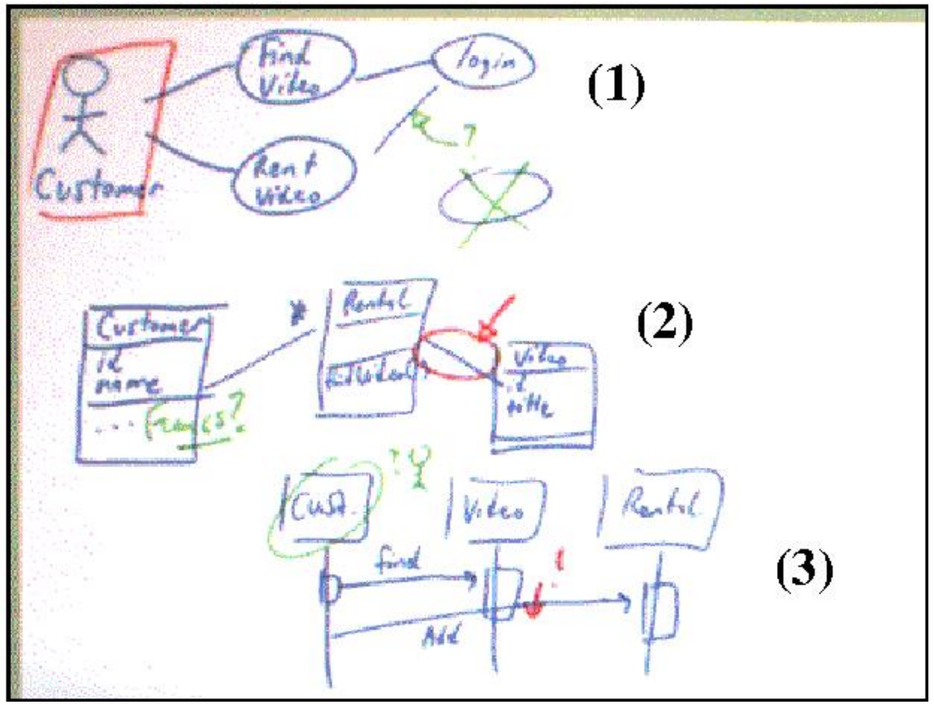
INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Uso mais comum com métodos ágeis
- Uso mais informal e leve da notação
- Objetivo não é ter um modelo completo
- A modelagem é usada para:
  - Conversar sobre uma parte do código ou do projeto
  - Documentar uma parte do código ou do projeto

Modelos de Software

- Como planta detalhada (blue print)
- Como esboços/rascunhos (sketches)

Modelagem com Esboço



Q. Chen, J. Grundy, J. Hosking: SUMLOW: early design-stage sketching of UML diagrams on an E-whiteboard. Software Practice and Experience, 2008

Modelagem com Esboço

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Modelagem são úteis em Engenharia Avante e em Engenharia Reversa

Engenharia Avante (Forward)

- Modelo é usado para discutir alternativas de projeto
- Antes de qualquer linha de código ser implementad

Engenharia Reversa

- Modelo é usado para explicar um código que já existe
- Contextos de manutenção e evolução de software

Modelagem de Domínio

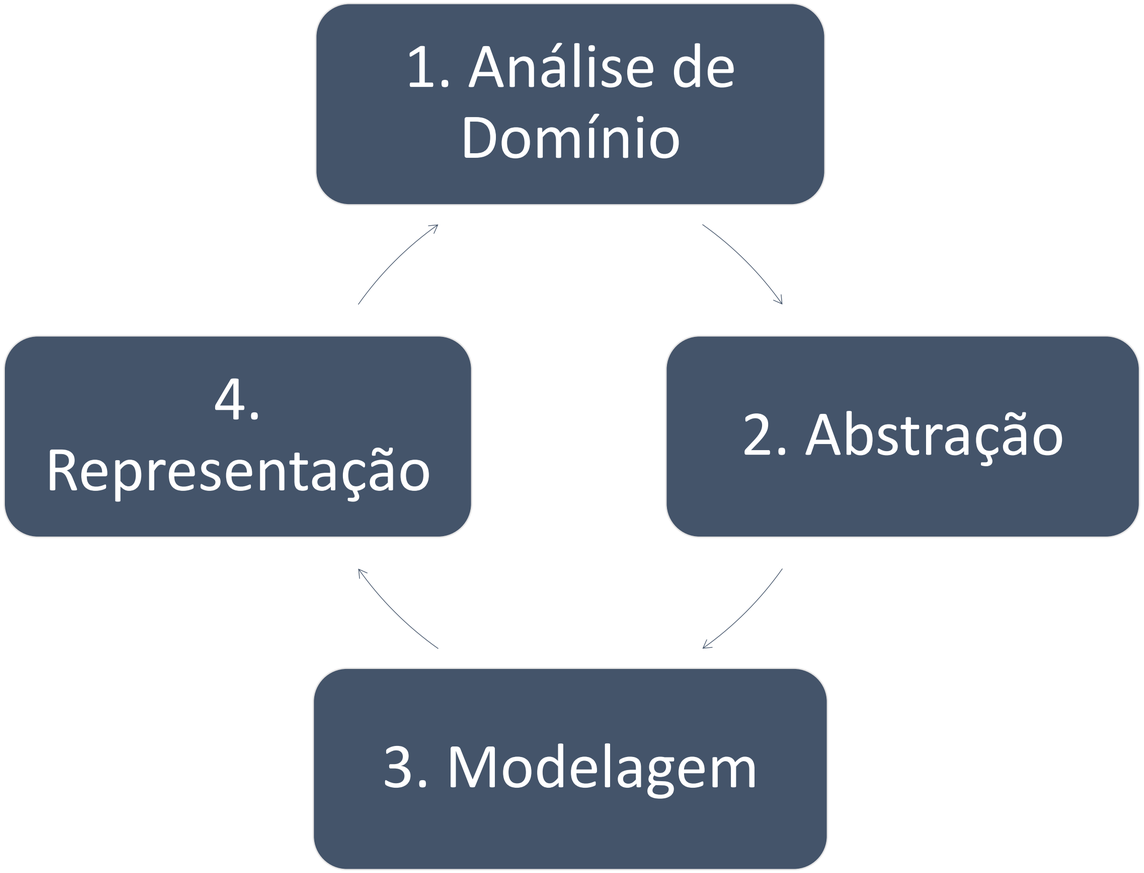
INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

1. Identificar o contexto do Sistema de Informação (Dados, Processo e Atores Envolvidos);

2. Idealizar como funcionará o Sistema de Informação pretendido;

3. Modelar a solução idealizada;

4. Representar o modelo idealizado numa linguagem clara e legível.



Como representar um Software?

A Linguagem de Modelagem Unificada, ou **UML (Unified Modeling Language),** foi criada através da colaboração de um grupo de pessoas influentes na área de engenharia de software nos anos 90.

Nos anos 90, a engenharia de software estava passando por uma fase de transição. As metodologias tradicionais estavam sendo questionadas e uma necessidade crescente de uma linguagem padrão para modelagem de sistemas de software estava surgindo. Havia várias metodologias de modelagem em uso, como Booch, OMT (Object Modeling Technique) e OOSE (Object-Oriented Software Engineering), cada uma com suas próprias notações e abordagens.

Como representar um Software?

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

A UML foi inicialmente desenvolvida como uma fusão de três metodologias principais:

Booch Method:

- Criado por Grady Booch, era uma das metodologias mais antigas e amplamente adotadas para modelagem de software.
- Focava em diagramas de classes, diagramas de estado, diagramas de sequência, entre outros.

Object Modeling Technique (OMT):

- Desenvolvido por James Rumbaugh, era outra abordagem popular para modelagem de sistemas orientados a objetos.
- Tinha um foco especial em diagramas de objetos, diagramas de estados, entre outros.

Object-Oriented Software Engineering (OOSE):

- Desenvolvido por Ivar Jacobson, foi outra metodologia influente com forte ênfase na análise e design orientados a objetos.
- Introduziu conceitos como casos de uso.

Como representar um Software?

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Unificação, Desenvolvimento e Evolução

Em 1995, Grady Booch, James Rumbaugh e Ivar Jacobson uniram forças para criar uma linguagem de modelagem padrão, unificando o melhor de suas respectivas metodologias. Eles foram apoiados por outras empresas e especialistas em engenharia de software, formando o "Three Amigos" (Três Amigos).

O objetivo era criar uma linguagem que pudesse abranger todos os aspectos do processo de desenvolvimento de software, desde a concepção até a implementação. A UML foi destinada a ser uma linguagem gráfica para visualizar, especificar, construir e documentar os artefatos de um sistema de software.

O trabalho inicial resultou na primeira versão da UML, conhecida como UML 0.8, em 1996. Ela foi seguida pela UML 0.9 e, finalmente, pela UML 1.0 em 1997, que foi amplamente adotada pela indústria de software. Desde então, a UML passou por várias revisões e atualizações. A mais recente versão 2.5 com uma ampla gama de diagramas e elementos para modelagem de sistemas complexos.

Como representar um Software?

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Contribuições e Aceitação

A UML se tornou a linguagem padrão para modelagem de software. Sua aceitação foi impulsionada por uma série de fatores:

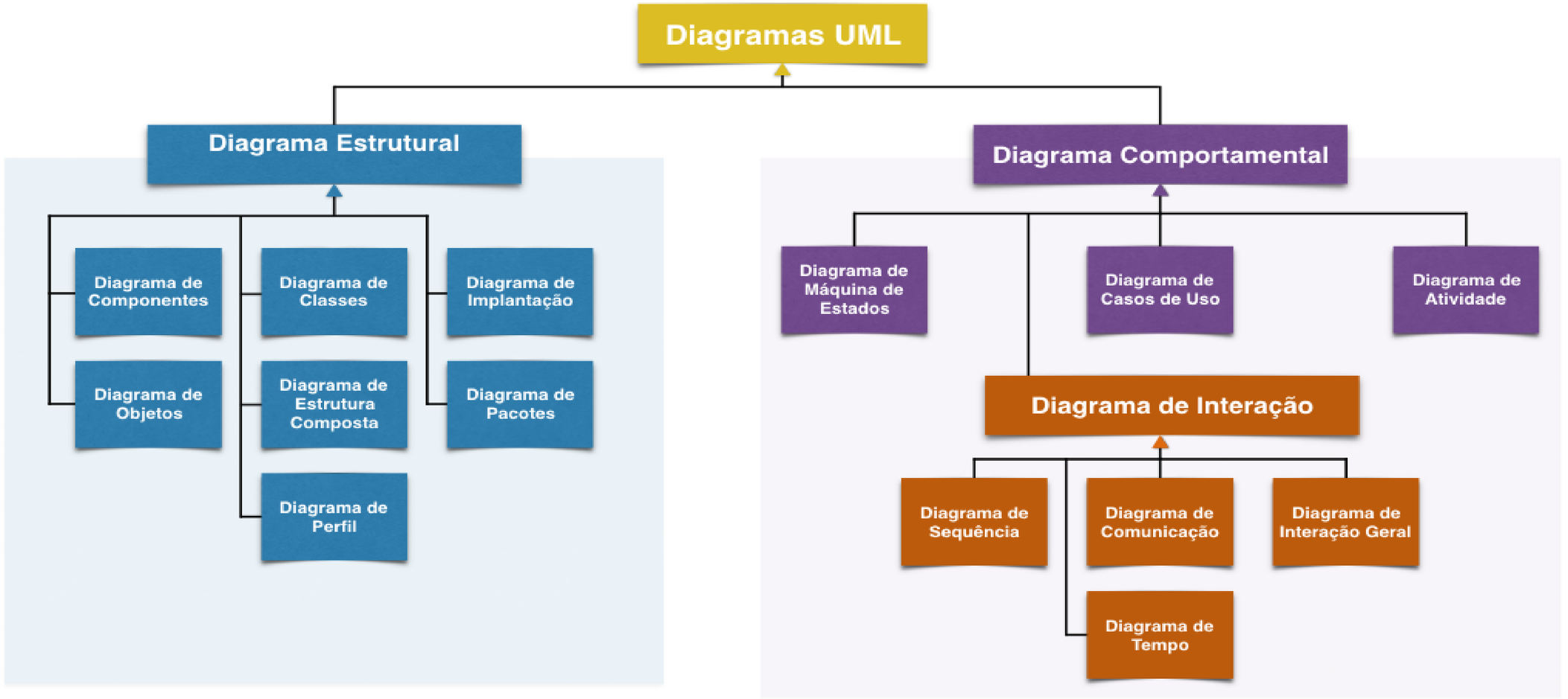
Padronização: Tornou-se um padrão da indústria, facilitando a comunicação entre equipes de desenvolvimento e organizações.

Flexibilidade: Oferece uma variedade de diagramas para atender às diferentes fases e aspectos do desenvolvimento de software.

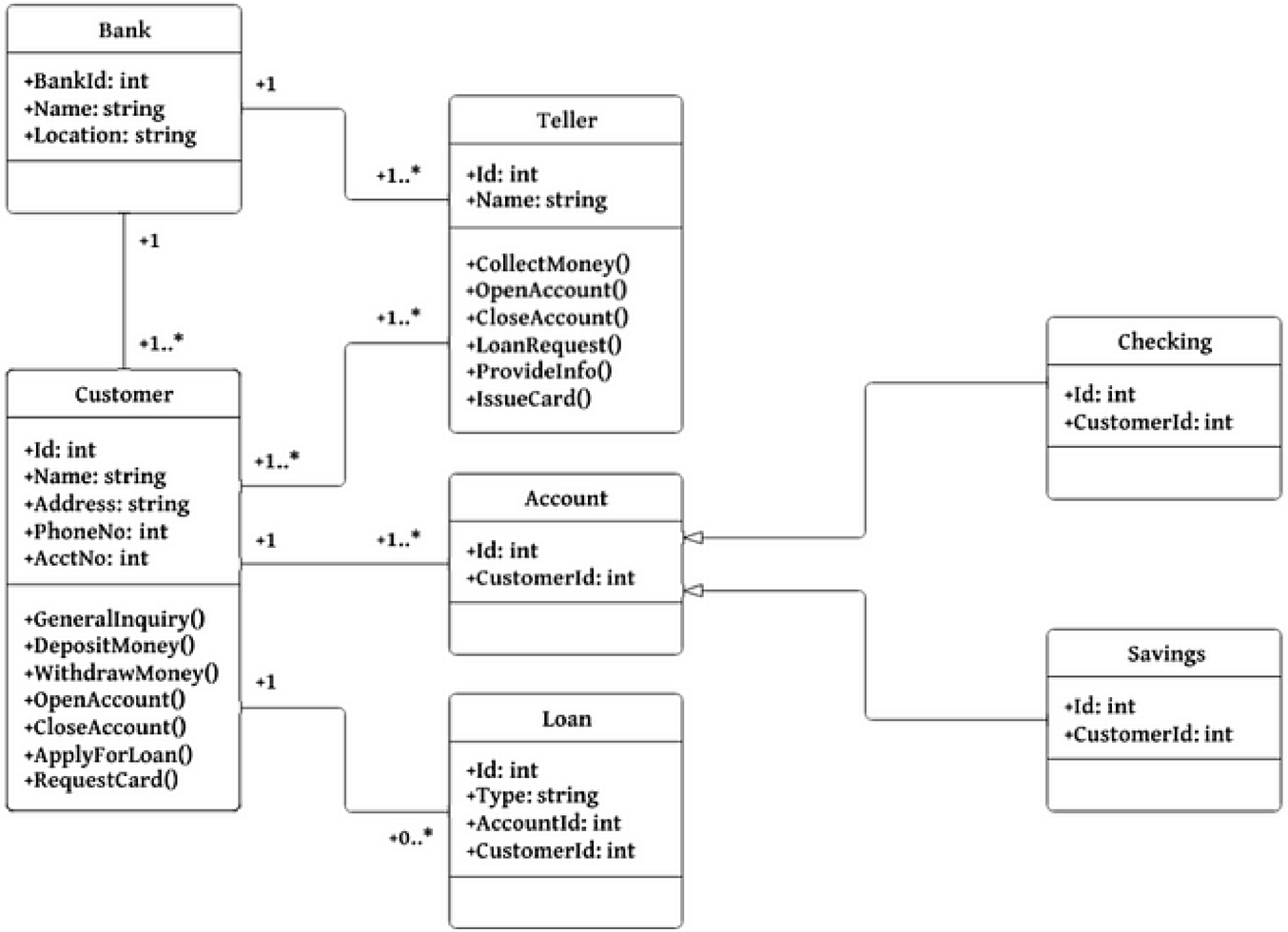
Ampla Comunidade: Conta com uma comunidade ativa de desenvolvedores, profissionais de engenharia de software e pesquisadores, que contribuem para seu desenvolvimento contínuo.

Ferramentas de Suporte: Há uma ampla variedade de ferramentas de modelagem UML disponíveis, tanto comerciais quanto de código aberto.

Diagramas UML



Diagramas de Classe



Programação Orientada a Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

A Orientação a Objetos (OO) é um paradigma de programação que se baseia na ideia de organizar o código em torno de objetos, que são instâncias de classes. É uma abordagem poderosa e amplamente utilizada no desenvolvimento de software, pois ajuda a criar sistemas **mais organizados, modulares e reutilizáveis**.

A Orientação a Objetos promove uma abordagem mais natural e organizada para modelar o mundo real em código, tornando-o mais legível, manutenível e flexível. É amplamente utilizada em linguagens de programação como Java, C++, Python, C#, entre outras, e é aplicável a uma variedade de domínios de desenvolvimento de software, desde aplicações desktop até sistemas distribuídos e aplicativos web.

Programação Orientada a Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Aqui estão os principais conceitos do paradigma de Orientação a Objetos:

Objeto: Um objeto é uma instância concreta de uma classe. Ele representa uma entidade do mundo real e possui atributos (propriedades) que descrevem seu estado e métodos que definem seu comportamento. Por exemplo, um objeto "Carro" pode ter atributos como "cor" e "velocidade" e métodos como "ligar" e "desligar".

Classe: Uma classe é um modelo ou uma definição abstrata que descreve as características comuns a um conjunto de objetos. Ela serve como um plano para a criação de objetos. A classe define os atributos e métodos que seus objetos terão em comum. Por exemplo, uma classe "Carro" define os atributos e métodos que todos os carros terão

Classes e Objetos

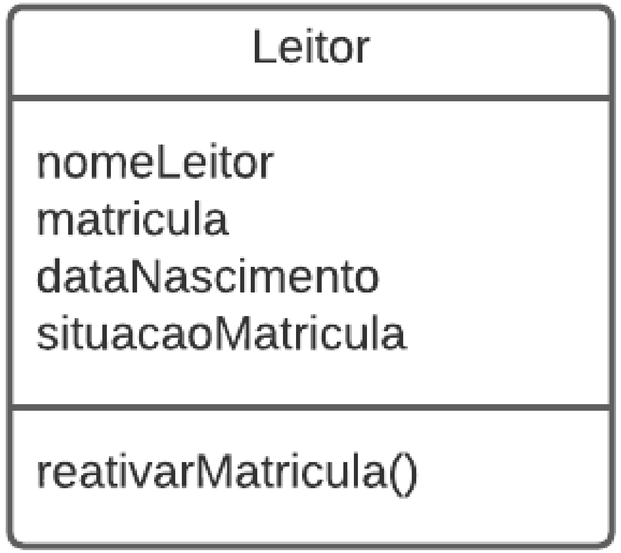
INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Modelo Conceitual x Diagrama de Classes

Classe

- É uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica.

- É representadas por um retângulo que pode possuir até três divisões:
  - Nome da classe
  - Atributos da classe
  - Métodos da classe



Classes e Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

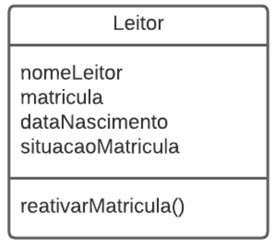
Qual é a diferença entre Classe e Objeto ?

Classe

- é definição do tipo;
- representa um conjunto de objetos de mesmo tipo;
- Classe = {obj1, obj2, obj3, ..., objN}

Objeto

- cada instância derivada da classe;
- é um elemento do conjunto representado pela classe



nomeLeitor: Maria
matrícula:20230001
dataNascimento: 15/10/2017
Situacao:Ativo

nomeLeitor: João
matrícula:20230002
dataNascimento: 10/08/2017
Situacao:Ativo

nomeLeitor: Lia
matrícula:20230003
dataNascimento: 20/12/2017
Situacao:Ativo

Classes – Atributos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Representa as características (campos ou dados) de uma classe;
- Exemplo: Leitor (nome, dataNascimento, sexo, email);
- No Diagrama de Classe também é permitido estabelecer:
  - A visibilidade de atributo (+ público, #protegido, ~pacote e- privado)
  - O tipo do dado do atributo (numérico, alfanumérico, data etc)
  - O valor pré-definido (inicial) do atributo;

Classes – Atributos (Visibilidade)

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Privado (-)
  - Somente a própria classe pode manipular o atributo
- Pacote (~)
  - Qualquer classe do mesmo pacote pode manipular o atributo
- Protegido (#)
  - Qualquer subclasse pode manipular o atributo
- Publico (+)
  - Qualquer classe do sistema pode manipular o atributo

Classes – Métodos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Representa atividades que um objeto de uma classe pode executar;
- Exemplo: Leitor (cadastrarLeitor());
- No Diagrama de Classe também é permitido estabelecer:
  - A visibilidade do método (+ público, #protegido, ~pacote e- privado)
  - O tipo do dado de retorno do método (numérico, alfanumérico, data etc)
  - Definir os parâmetros dos métodos;

Classes – Métodos (Visibilidade)

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Privado (-)
  - Funcionalidades de apoio à própria classe
- Pacote (~)
  - Funcionalidades de apoio a outras classes do pacote
- Protegido (#)
  - Funcionalidades que precisam ser estendidas por outras classes
- Publico (+)
  - Funcionalidades visíveis por todas as classes do sistema

Diagrama de Classes

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Classe

- É uma abstração que descreve um grupo de objetos com as mesmas propriedades (atributos), comportamento (operações), tipos de relacionamentos e semântica

Diagrama de Classe

- Apresenta uma visão estática de como as classes estão organizadas a fim de definir sua estrutura lógica

Diagrama de Classes

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

O que podemos descrever no diagrama de classes?

- As classes em si
- Seus atributos (campos, dados, propriedades...)
- Seus métodos (operações, transações...)
- Os relacionamentos entre as classes
- Os papéis das classes
- A multiplicidade envolvidas nos relacionamentos

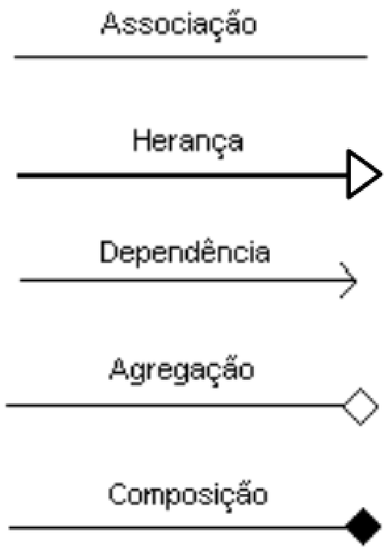
D. de Classes - Relacionamentos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Permite compartilhar informações e colaborar com a execução dos processos do sistema.

- Descreve um vínculo que ocorre, normalmente, entre os objetos de uma ou mais classes.

- Os tipos de relacionamento são:
  - Associação
    - Agregação
    - Composição
  - Especialização/Generalização
  - Dependência



D. de Classes - Relacionamentos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Associação

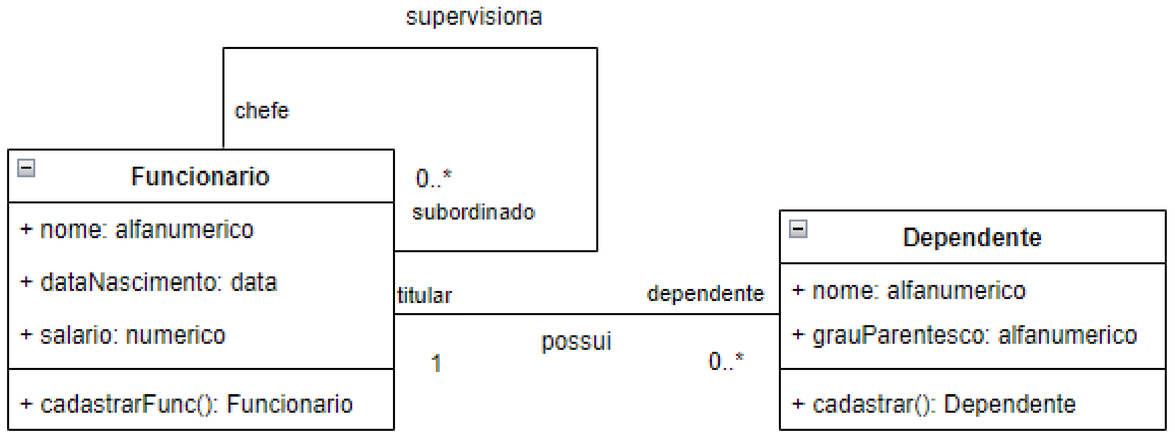
- Descreve um conjunto de vínculos entre elementos de modelo.
- Relacionamento estrutural que especifica objetos de um item conectados a objetos de outro item:
  - Associação binária – quando há duas classes envolvidas na associação de forma direta de uma para outra.
    - Relacionamento entre duas classes (tipo mais comum).
    - Podem possuir títulos para determinar o tipo de vínculo.
  - Associação unária – quando há um relacionamento de uma classe consigo mesma.

D. de Classes - Relacionamentos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Associação Unária (ou reflexiva)

- Ocorre quando há um relacionamento de um objeto de uma classe com objetos da mesma classe;
- No exemplo abaixo, percebe-se que um objeto da classe Funcionário pode (ou não) supervisionar outros objetos dessa mesma classe;
- Para o relacionamento ficar mais claro, pode-se informar a sua multiplicidade.



D. de Classes - Relacionamentos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Multiplicidade

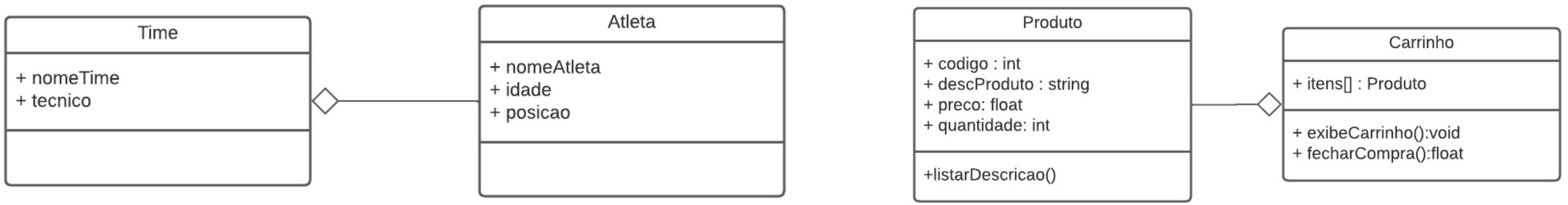
Multiplicidade	Significado
0..1	No mínimo zero e no máximo um. Os objetos não precisam estar relacionados, porém se houver relacionamento deve ser de no máximo 1
1..1 (1)	Um e somente um (valor default)
0..*	No mínimo nenhum e no máximo muitos
*	Muitos
1..*	No mínimo um e no máximo muitos
3..5	No mínimo 3 e no máximo 5

D. de Classes - Relacionamentos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Agregação

- Tipo especial de associação que tenta demonstrar que as informações de um objeto-todo precisam ser complementadas pelas informações contidas em um (ou mais) objetos-parte.
- A existência do objeto-parte faz sentido mesmo não existindo o objeto-todo.
- A associação de agregação pode, em muitos casos, ser substituída por uma associação binária simples, dependendo da visão de quem faz a modelagem.

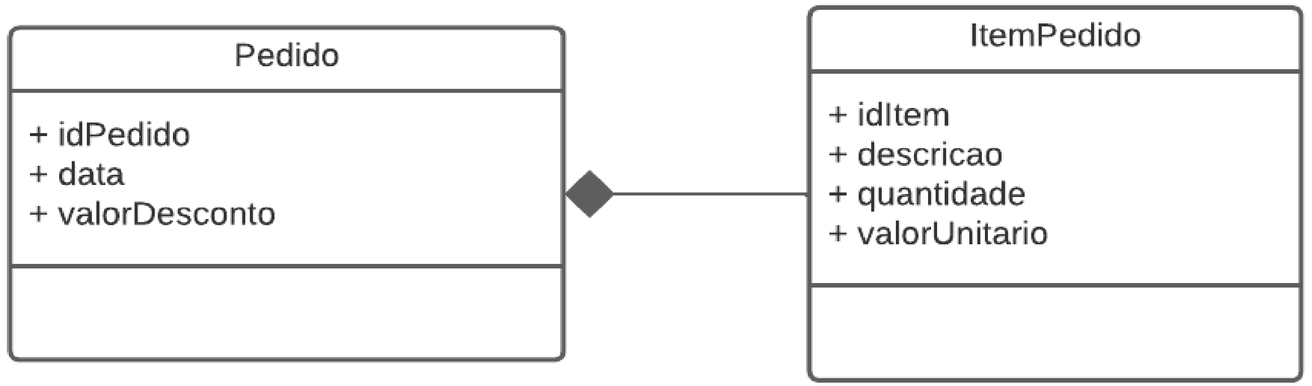


D. de Classes - Relacionamentos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Composição

- É uma variação da agregação e considerada mais “forte”.
- O objeto-parte não pode existir sem o objeto-todo.
- Se o objeto-todo for destruído, o objeto-parte também será.

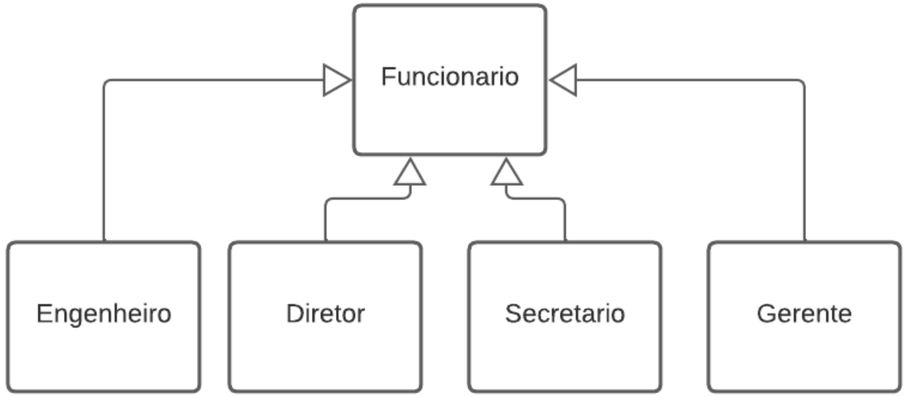


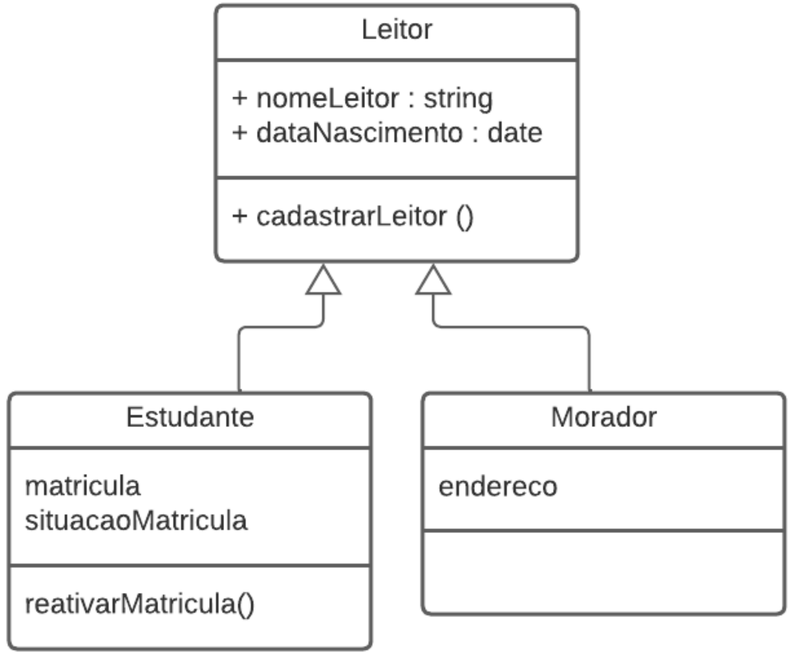
D. de Classes - Relacionamentos

Especialização/Generalização

- Tem como objetivo identificar classes-mãe, denominadas de gerais, e classes-filha chamadas de especializadas;

- São chamados de relacionamentos “é um tipo de".



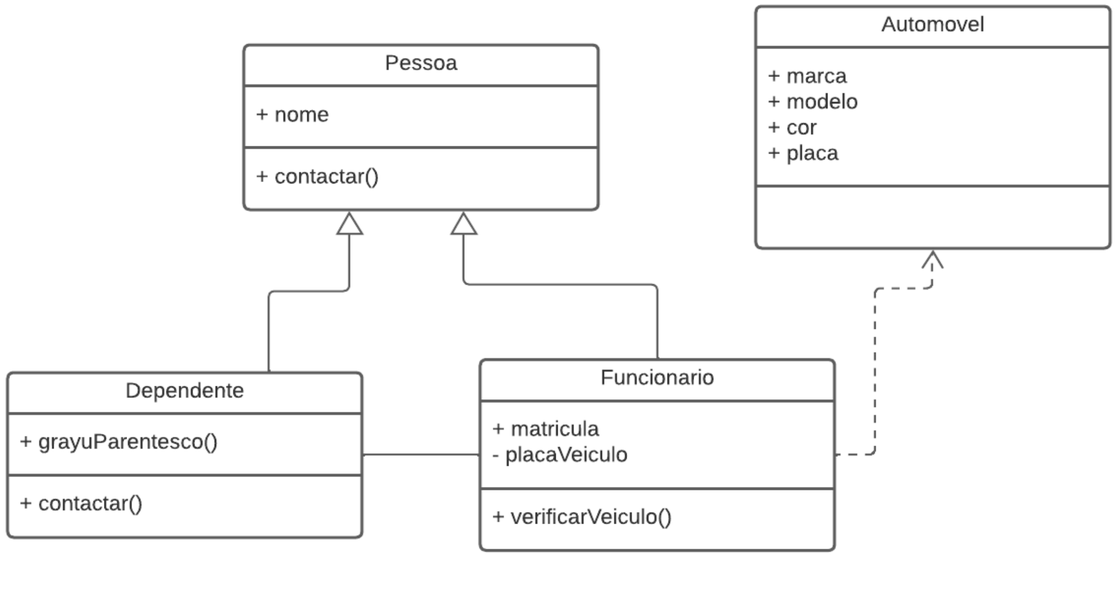


D. de Classes - Relacionamentos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Dependência

- Como o nome sugere, indica um grau de dependência entre uma classe e outra.
- Uma dependência difere de uma associação porque a conexão entre as classes é temporária.
- Representada por uma seta tracejada entre duas classes.

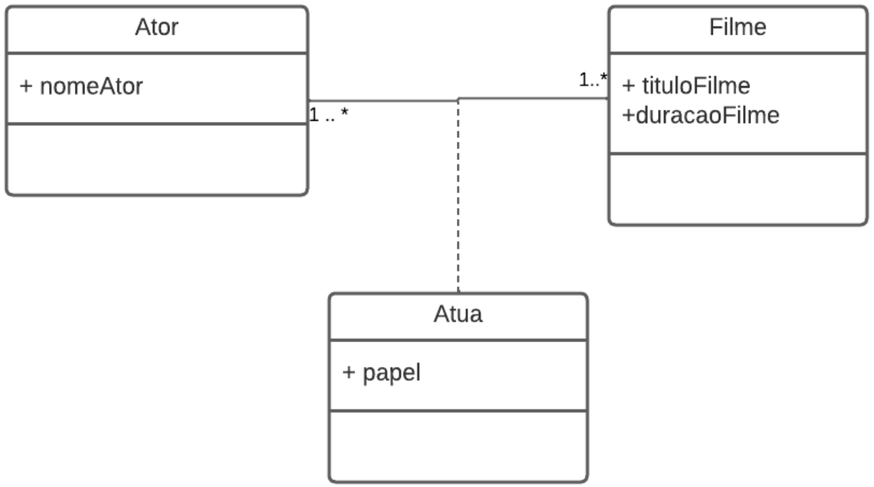


D. de Classes - Relacionamentos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Classe Associativa

- Utilizada quando ocorrem associações que possuem multiplicidade muitos para muitos em todas as suas extremidades;
- Armazena os atributos transmitidos pela associação;
- Pode possuir seus próprios atributos;
- Representada por uma reta tracejada partindo do meio da associação até uma classe.



D. de Classes - Relacionamentos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Classe Intermediária

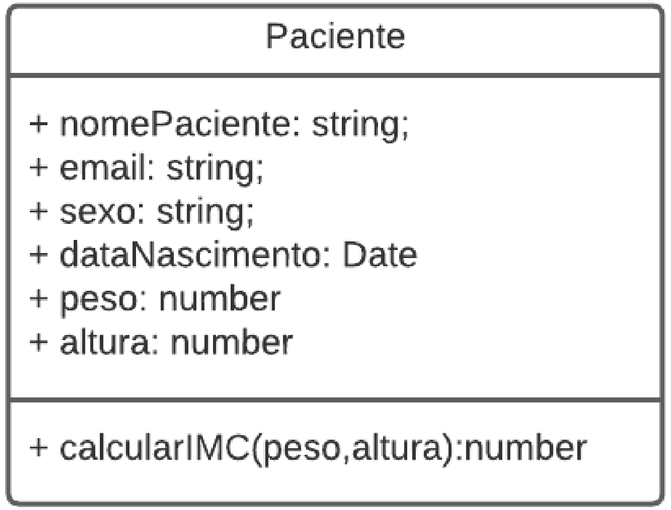
- Substitui as classes associativas;
- Apresenta, exatamente, a mesma função da classe associativa;.
- Pode possuir seus próprios atributos;

Representando o 1º Objeto

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Representando um objeto a partir de um Diagrama de Classe

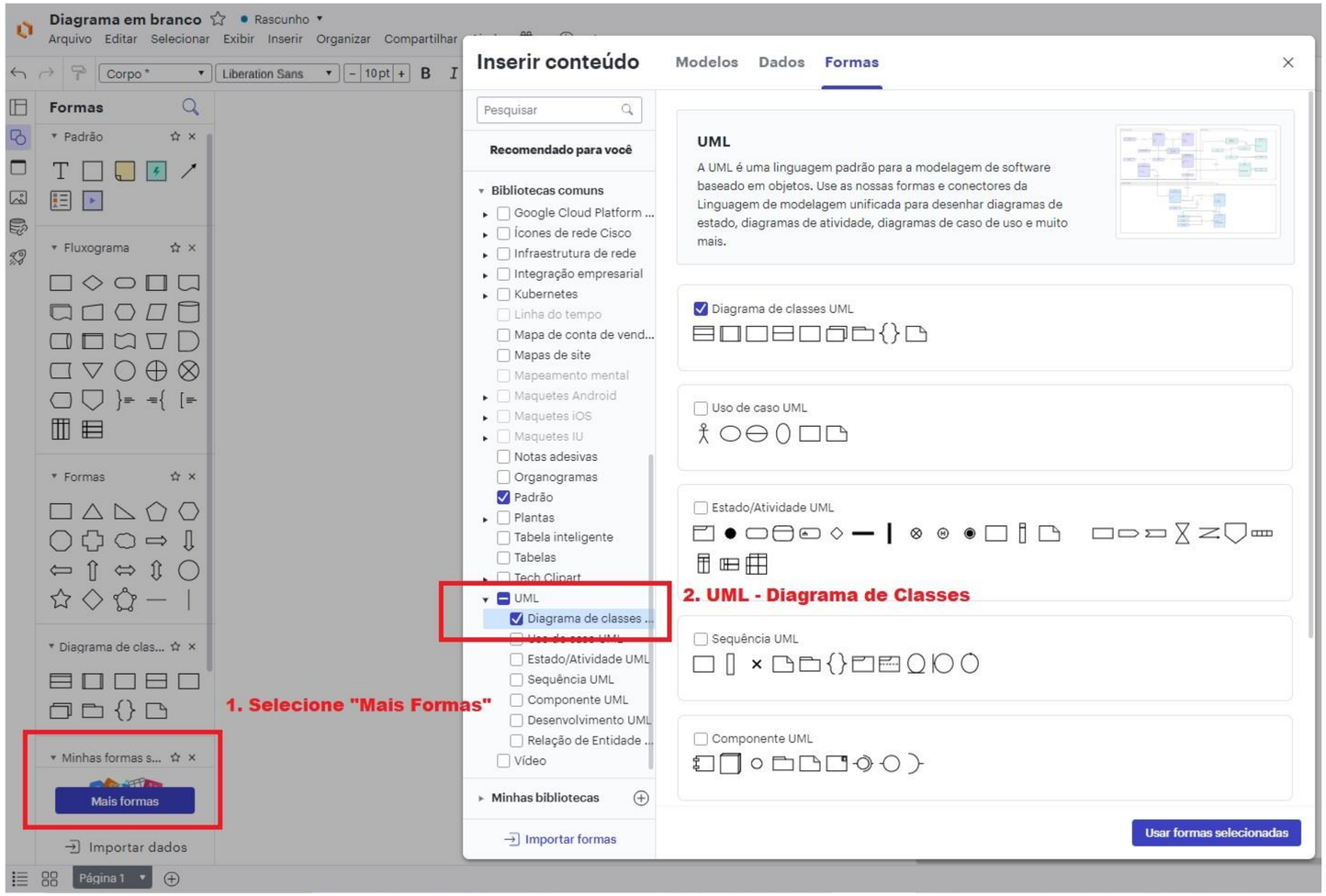
- Vamos utilizar a ferramenta LucidChart ou Diagrams (Draw.io)



Representando o 1º Objeto

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Acesse o link do aplicativo LucidChart disponível no site do professor
- Inclua o UML – Diagrama de Classes



Notações

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Existem várias notações para nomes de variáveis, métodos, classes, e outros elementos em programação. Cada uma dessas notações segue regras específicas sobre como as palavras devem ser formatadas e combinadas.

- Benefícios das Notações:

  - Legibilidade: Notações bem escolhidas facilitam a leitura e compreensão do código.
  - Convenção: Seguir uma convenção de nomenclatura ajuda na consistência do código e na colaboração em equipe.
  - Compatibilidade: Algumas linguagens ou ambientes têm convenções de nomenclatura específicas que devem ser seguidas para compatibilidade e boas práticas.

- Em resumo, a escolha da notação de nomenclatura adequada depende das convenções da linguagem de programação, ambiente de desenvolvimento e práticas recomendadas. O importante é manter consistência dentro do projeto ou equipe para garantir um código claro e fácil de entende.

- Acesso a página do professor para conhecer as principais notações.

Notações


(underscores)
SCREAMING_SNAKE_CASE
(UPPER_SNAKE)
(dash)
kebab-case
(words-on-a-skewer)
(underscore)
snake_case
(lower_snake)
first word lower
capitalize all Following Words
camelCase
See also: UpperCamel
aka PascalCase

SI Biblioteca – Atividade Prática

1. Fazer cadastro no site do LucidChart;
2. Criar um novo diagrama utilizado a biblioteca de formas "UML – Diagrama de Classes";
3. Criar a classe Paciente com os atributos e o método;
4. Utilizar a convenção de nomenclatura para Javascript (veja este artigo)

Dúvidas?

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Dúvidas?

fernando.alves@ifrj.edu.br

ftamberlini.dev.br

Referências:VALENTE, M. T (2020). Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade, Editora: Independente.