

# Modelagem de Sistemas

Guilherme Henrique Pasqualin Algeri

[guilherme.algeri@sistemafiep.org.br](mailto:guilherme.algeri@sistemafiep.org.br)

# Diagrama de Classe

# Diagrama de Classe

Os conceitos da orientação a objetos surgiram da necessidade em se enfatizar unidades discretas, e obter a reutilização de código, mantendo-se a qualidade do software

# Diagrama de Classe

O núcleo do pensamento OO predomina num foco sobre os dados, em vez dos processos, compondo módulos auto-suficientes — os objetos

# Diagrama de Classe

Encerrando em sua estrutura todo o conhecimento dos dados e dos processos para manipulação desses dados

# Diagrama de Classe

Os conceitos fundamentais de orientação a objetos são o contrato que estabelece toda e qualquer implementação que se diga OO

# Diagrama de Classe

Sendo assim, se um desses conceitos não for atendido, não podemos afirmar que determinada tecnologia possa ser nomeada como orientada a objetos

# Diagrama de Classe

Vejamos então algumas definições da orientação a objetos:



# Diagrama de Classe

**Objeto:** um objeto é qualquer coisa existente no mundo real, em formato concreto ou abstrato, ou seja, que exista fisicamente ou apenas conceitualmente, e o qual se pode caracterizar e identificar comportamentos

# Diagrama de Classe

**Atributo:** as características associadas aos objetos são chamadas de atributos

# Diagrama de Classe

**Operação x Método:** o comportamento dos objetos é representado pelas operações. Contudo, a operação para um objeto representa apenas a definição do serviço que ele oferece a outras estruturas

# Diagrama de Classe

Quando tratamos da implementação dessa operação, ou seja, da sua representação em código, estamos nos referindo ao seu método

# Diagrama de Classe

**Mensagem:** é a solicitação que um objeto faz a outro, invocando a execução de um determinado serviço

# Diagrama de Classe

**Encapsulamento:** o conceito de encapsulamento nos remete ao fato de que a utilização de um sistema orientado a objetos não deve depender de sua implementação interna, e sim de sua interface

# Diagrama de Classe

Isso garante que os atributos e os métodos estarão protegidos, só podendo ser acessados pela interface disponibilizada pelo objeto, ou seja, sua lista de serviços

# Diagrama de Classe

**Herança:** ao refinarmos a modelagem de um sistema é comum encontrarmos características redundantes entre objetos. Essa redundância pode ser evitada pela separação dos atributos e operações numa classe comum, identificada como superclasse



# Diagrama de Classe

Essa classe comum (superclasse) passa a ser a generalização de outras classes que encerram em si apenas os atributos e operações específicos a cada uma

# Diagrama de Classe

**Polimorfismo:** uma operação pode ter implementações diferentes em diversos pontos da hierarquia de classes. Isso significa que ela terá a mesma assinatura (mesmo nome, lista de argumentos e retorno), porém implementações diferentes (métodos). Isso é o polimorfismo

# Elaboração do diagrama de classes

# Diagrama de Classe

Um diagrama de classes descreve os “tipos” de objetos do software e os vários tipos de relacionamentos estáticos que existem entre eles

# Diagrama de Classe

Este é o diagrama mais comumente utilizado por empresas que desenvolvem software orientado a objetos. Sua elaboração é uma atividade complexa

# Diagrama de Classe

Para simplificar nossa discussão, trataremos da construção do diagrama de classes em nível de domínio

# Diagrama de Classe

Neste caso, o objetivo é considerar apenas as principais entidades envolvidas no problema, como elas se relacionam, os atributos que as caracterizam, e as operações que são realizadas sobre cada entidade ou que cada entidade realiza

# Diagrama de Classe

Ou seja, consideraremos apenas os principais conceitos do domínio do problema e descartaremos aspectos relacionados à tecnologia que será utilizada em sua implementação



# Diagrama de Classe

Sendo assim, estaremos preocupados com a identificação de quatro tipos de elementos: as entidades (classes) do software, seus atributos, suas operações e o relacionamento entre elas

# Diagrama de Classe

Podemos utilizar uma metodologia para criação do modelo de classes dividida em cinco etapas, como pode ser visto na Figura 1

# Diagrama de Classe



# Identificar Classes

# Identificar Classe

Classes representam entidades (conceitos) do mundo real, ou seja, do problema para o qual o software estiver sendo desenvolvido

# Identificar Classe

Elas são uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica e representam o principal bloco de construção de um sistema orientado a objetos

# Identificar Classe

Para identificá-las, devemos utilizar nosso conhecimento sobre o domínio do problema e a especificação de requisitos do software elaborada

# Identificar Classe

Mais especificamente, procuramos na especificação de requisitos por conceitos que representem objetos do mundo real



# Identificar Atributos

# Identificar Atributos

Tendo definidas as classes, estamos preparados para especificar quais serão seus atributos

# Identificar Atributos

É importante deixar claro aqui que dividimos as atividades de identificação de classes e atributos em duas etapas por questões didáticas

# Identificar Atributos

É comum realizarmos as duas atividades em paralelo, quando já estamos acostumados a elaborar diagrama de classes

# Identificar Atributos

Neste contexto, podemos definir atributos como sendo as propriedades que caracterizam objetos definidos através das classes

# Identificar Atributos

Assim, para identificarmos os atributos das classes, devemos procurar por propriedades que caracterizam o objeto em questão, considerando a especificação dos requisitos e nosso conhecimento do domínio

# Identificar Atributos

Outro aspecto importante que devemos ter em mente quando estamos definindo os atributos é sua visibilidade

# Identificar Atributos

Este aspecto indica se o atributo poderá ser acessado diretamente por outra classe que esteja relacionada com a classe onde se encontra o atributo ou se será um atributo para uso interno da própria classe



# Identificar Atributos

É uma boa prática definir todos os atributos como privados e para cada um deles definir métodos de acesso

# Identificar Atributos

Dessa forma, escondemos as informações de um objeto e fornecemos acesso a elas indiretamente através de operações

# Identificar Atributos

Fazendo isto, estamos trabalhando com um dos conceitos básicos da orientação a objetos; o encapsulamento. Isto proporciona maior controle sobre o estado do objeto

# Identificar Operações

# Identificar Operações

O próximo passo é a definição de quais operações estarão alocadas em cada classe. As operações definem as habilidades de um determinado objeto, ou seja, tudo aquilo que ele pode realizar (ações)

# Identificar Operações

Dessa forma, ao procurar por operações estaremos identificando ações que o objeto de uma classe é responsável por desempenhar no contexto do sistema

# Identificar Operações

Assim como fizemos com os atributos, outro aspecto importante que devemos ter em mente quando estamos definindo as operações é sua visibilidade

# Identificar Operações

Esta indica se a operação poderá ser acessada diretamente por outra classe que esteja relacionada com a classe onde se encontra a operação ou se será uma operação para uso interno da própria classe



# Identificar Relacionamentos

# Identificar Relacionamentos

Com todas as classes identificadas e especificadas, devemos definir como elas estão relacionadas para que possamos elaborar o diagrama de classes

# Identificar Relacionamentos

Os principais tipos de relacionamento entre classes presentes na UML são:

# Identificar Relacionamentos

**Associação:** são relacionamentos estruturais entre instâncias e especificam que objetos de uma classe estão ligados a objetos de outras classes

# Identificar Relacionamentos

É representado por uma linha simples ligando duas classes e pode conter atributos como nome, multiplicidade e navegabilidade

# Identificar Relacionamentos

O nome descreve a semântica (significado) do relacionamento; a multiplicidade indica quantos objetos de uma determinada classe um objeto pode se comunicar

# Identificar Relacionamentos

a navegabilidade especifica quem enxerga quem no relacionamento, ou seja, se um objeto tem conhecimento da existência de outro

# Identificar Relacionamentos

**Generalização (herança simples ou composta):** relacionamento entre um elemento mais geral e um mais específico onde o elemento mais específico herda as propriedades e operações do elemento mais geral



# Identificar Relacionamentos

A relação de generalização também é conhecida como herança na orientação a objetos e denotam relações “é um tipo de” onde subclasses são especializações de superclasses

# Identificar Relacionamentos

**Agregação regular:** tipo de associação (é parte de, todo/parte) onde o objeto parte é um atributo do todo. Agregação é representada por uma linha simples com um losango sem preenchimento do lado do todo

# Identificar Relacionamentos

**Composição:** relacionamento entre um elemento (o todo) e outros elementos (as partes) onde as partes só podem pertencer ao todo e são criadas e destruídas com ele

# Identificar Relacionamentos

O relacionamento de composição é representado por uma linha simples com um losango preenchido do lado do todo

# Identificar Relacionamentos

Para identificarmos esses tipos de relacionamentos, aliado a nosso conhecimento no domínio do problema, efetuamos a leitura da especificação dos casos de uso e, para relacionamentos de:

# Identificar Relacionamentos

**Generalização:** verificamos se há alguma relação “é um tipo de” entre as classes identificadas;

# Identificar Relacionamentos

**Associação:** verificamos se há a necessidade de um objeto de uma classe utilizar serviços disponibilizados por um objeto de outra classe ou, simplesmente, conhecer o outro objeto

# Identificar Relacionamentos

**Agregação:** verificamos se há alguma relação “parte de” entre as classes identificadas;



# Identificar Relacionamentos

**Composição:** verificamos se há alguma relação “parte de” forte entre as classes identificadas

# Revisar Modelo

# Revisar Modelo

Nesta etapa, devemos revisitar o modelo construído e verificar se ele faz sentido e contempla os conceitos definidos na especificação de requisitos do software. Identificando-se algum problema, este deve ser ajustado

# Revisar Modelo

Estando o diagrama finalizado, este poderá ser entregue para a equipe de desenvolvimento junto com o arquiteto de software que realizará os devidos ajustes para tornar o diagrama mais próximo daquilo que será realmente implementado

# Revisar Modelo

Ou seja, inserir aspectos tecnológicos no diagrama de domínio. Feito isto, teremos nosso projeto detalhado (também conhecido como diagrama de classes de baixo nível)

# Exemplo prático

# Exemplo Prático

Exemplo em Anexo

# Obrigado!

Guilherme Henrique Pasqualin Algeri  
[guilherme.algeri@sistemapiep.org.br](mailto:guilherme.algeri@sistemapiep.org.br)  
(42) 9 9148-8117