ANÁLISE E PROJETO ORIENTADO A OBJETOS

Professores:

Ana Paula Gonçalves Serra

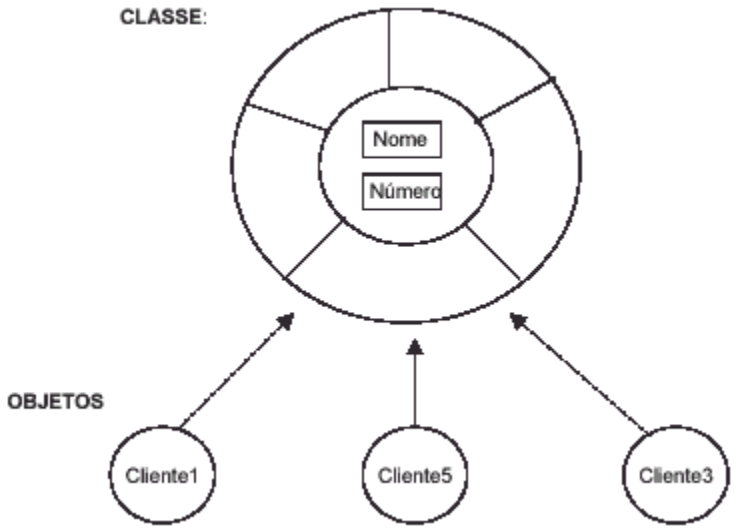
André Luiz Ribeiro

Keity Yamamoto

Diagrama de Classes – Parte I

1. Descrição

O Diagrama de Classes é o principal digrama da UML, sendo estático, modela a estrutura do sistema, através de atributos, operações e relacionamentos. Uma classe é a descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica.



2. Aplicação

Utilizado para:

- Propor a estrutura do sistema.
- Auxiliar na definição de componentes e na implementação.

Palavras chaves: Estrutura do sistema.

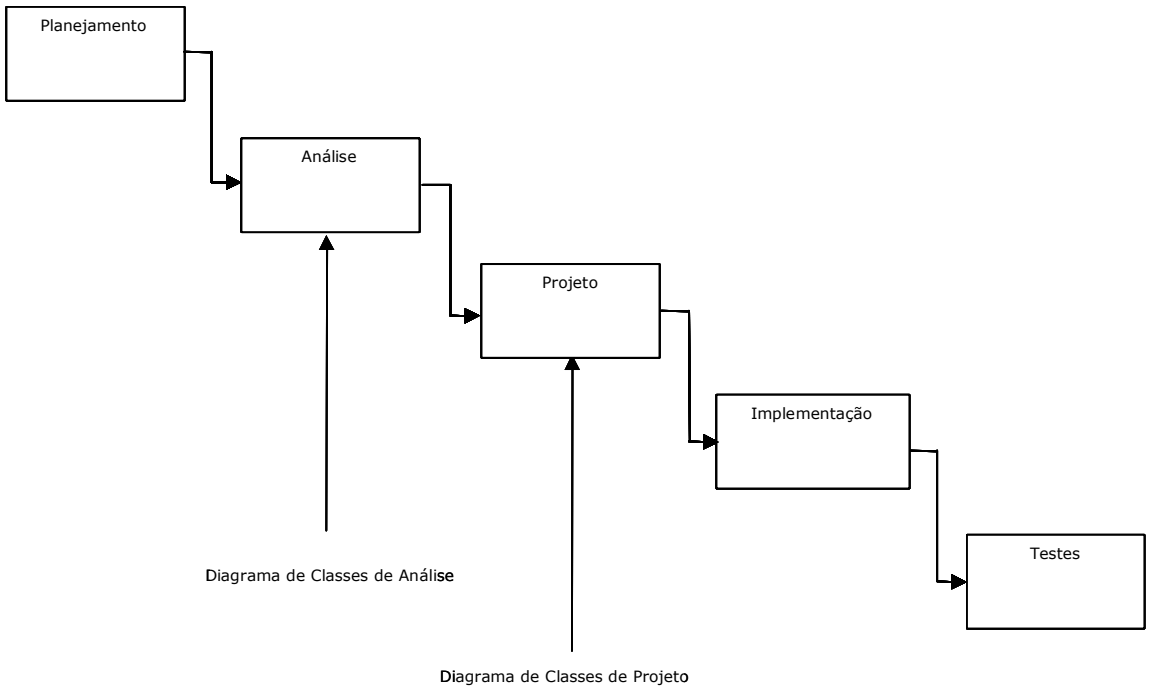
O diagrama de Classes é um dos produtos da fase de Análise e/ou Projeto e pode ser elaborado através de ferramentas CASE. O diagrama de classes é o fundamento da modelagem orientada a objetos auxiliando em toda modelagem, definição dos componentes e codificação.

O diagrama de classes pode ser classificado em dois tipos:

- **Diagrama de Classes de Análise/Negócio:** Representam as classes de negócio do sistema, sendo um primeiro modelo conceitual para representar elementos do sistema que possuam características, ações e responsabilidades. Com o "tempo", as classes de análise evoluem e se transformam em classes de projeto. As classes de análise não devem se preocupar diretamente com a implementação, ou seja, com particularidades da linguagem de programação.

- **Diagrama de Classes de Projeto/Implementaç:** Representam as classes de negócio e de implementação do sistema. Para isso, o diagrama de classes de análise deve ser mapeado para um diagrama de projeto que deve abranger as classes de análise, componentes de implementação, interfaces externas, particularidades relevantes da linguagem de programação e da arquitetura do sistema. **Esse diagrama é elaborado com base na "Realização dos Casos de Uso".**

Obs: Alguns autores Scott e Fowler, classificam os diagramas de classes em 3 tipos: Conceitual, para definir as classes de negócio. Especificação, para definir as interfaces. Implementação: para definir as classes de implementação. Para nossas aulas o diagrama conceitual equivale ao diagrama de classes de análise, e o diagrama de especificação e implementação serão desenvolvidos em um único que será o diagrama de classes de projeto.

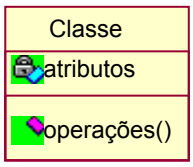


3. Notação Básica – Diagrama de Classes de Análise/Negócio

- **CLASSE X OBJETO**
  - **Classe**
    - Representam algo do mundo real;
    - Define as características e o comportamento dos objetos.
  - **Objeto**
    - Uma ocorrência de uma classe;
    - Criados durante a execução do sistema

- **CLASSE**
  - Classe é um conjunto de objetos que possuem características e comportamentos comuns;
  - Podem ser qualquer elemento que seja relevante dentro do domínio do problema;
  - Devem Iniciar com letra maiúscula e no singular;
  - Podem ter nomes compostos.

Uma classe é representada por um retângulo composto de Nome, Atributos e Operações.



Nome da classe: o nome da classe deve expressar o que a classe representa, pode ser um texto (substantivo) simples ou composto no singular e iniciado com letra maiúscula.

Atributos: um atributo é uma propriedade (características) da classe. Em determinado momento, um objeto de uma classe terá um valor específico para cada um dos atributos de sua classe.

- Descrevem as características dos objetos;
- Definem os dados que devem ser armazenados;
- São as informações que uma classe deve ter de si mesma;
- Exemplo Classe Cliente: nome, cnpj, inscrição estadual.

Operações: uma operação é uma ação ou transformação realizada ou sofrida por um objeto. Muitas vezes utiliza-se o termo método ao invés de operação. Método é a implementação de uma operação para uma classe.

- São utilizados para:
  - Manipular os atributos

- Realizar outros tipos de ações.

- Pertencem às classes e somente podem ser aplicados aos objetos da classe;
- Definem os serviços que a classe pode oferecer.

▪ RELACIONAMENTOS

Um relacionamento é uma conexão entre classes, que representa comunicação e interação entre classes.

Os tipos de relacionamentos podem que podem ser usados são associação (simples, agregação e composição), generalização e dependência.

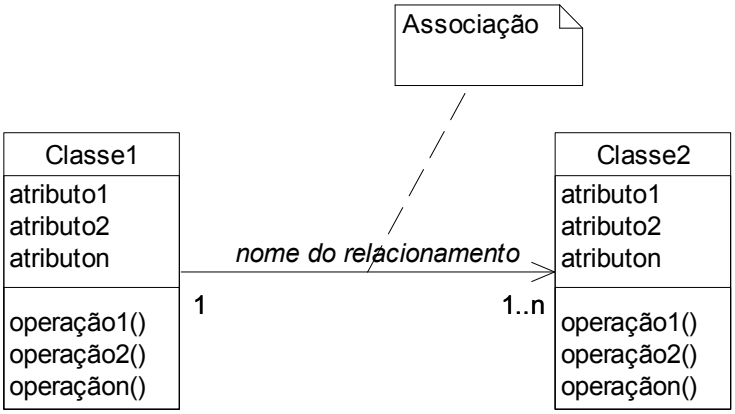
Associação: é um relacionamento que demonstra que uma classe está conectada a outra, com o intuito de comunicação entre elas. A associação normalmente é bidirecional, isso significa que se um objeto é associado a outro objeto, ambos os objetos se "conhecem". A associação simples é uma conexão entre classes. Isto é representado por uma linha contínua entre as duas classes. A associação deve possuir um nome, representado próximo a linha de associação, frequentemente um verbo. Também é indicado que a associação possua uma seta, que indica a direção de leitura. Tanto o nome da associação como a seta é importante para facilitar o entendimento do domínio do problema

Na associação deve-se especificar a cardinalidade entre as classes. A cardinalidade é a indicação de quantos objetos podem participar de um relacionamento, ou seja, multiplicidade. A multiplicidade das associações define todas as possibilidades numéricas de ocorrências entre os objetos.

Exemplos de multiplicidades:

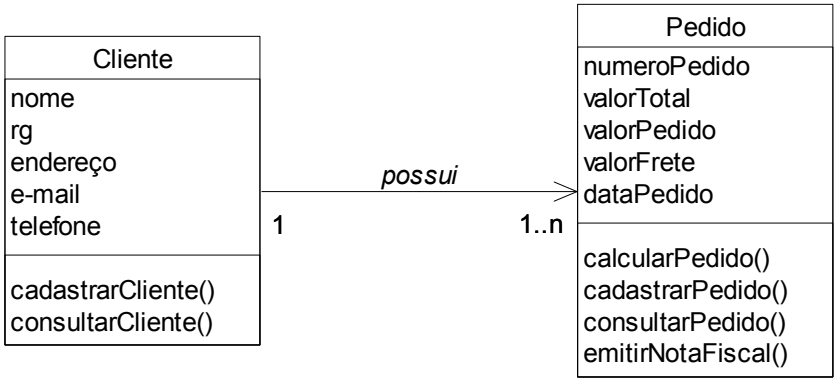
zero a um	(0..1)
zero a muitos	(0..*) ou (0..n)
um a muitos	(1..*) ou (1..n)
sem indicação	1
3 exato	(3)
1 a 3	(1..3)
série	(1,2,5..8)

A associação e cardinalidade entre as classes está diretamente relacionada ao domínio do problema.



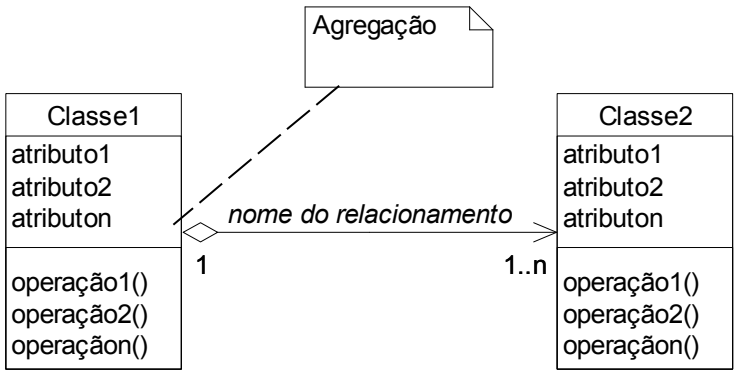
Exemplo:

O exemplo a seguir representa que um cliente possui 1 ou n pedidos e que um pedido pertence a somente um cliente.



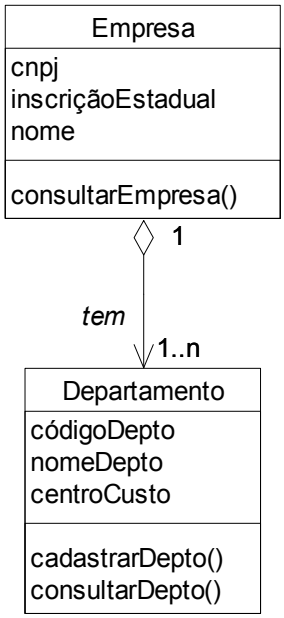
Além da associação simples, apresentada acima, existem dois tipos específicos de associação:

Agregação: é um caso especial de associação, que representa conceitualmente que uma ou mais classes fazem parte de uma outra classe (“todo parte”). Mas as classes “todo parte” podem ”viver” independentes. A notação da agregação é um losango sem preenchimento.



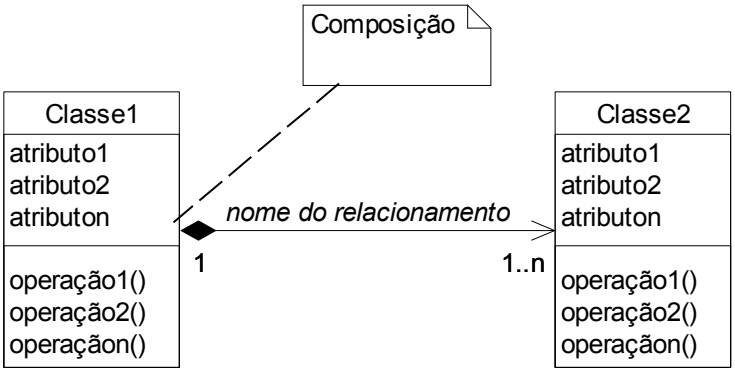
Exemplo:

O exemplo a seguir representa que uma empresa pode ser composta de 1 ou mais departamentos. Sendo que não necessariamente ao instanciar a classe Empresa a classe Departamento deve ser instanciada.



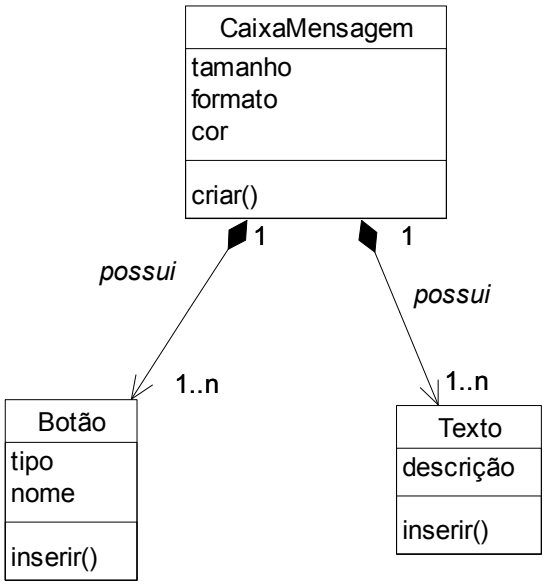
time → atletas

Composição: é um caso especial de associação e uma forma de agregação mais forte, onde o objeto parte depende do objeto todo, ou seja, espera-se que as partes "vivam" e "morram" como um todo conjuntamente. A notação da agregação em um losango preenchido.



Exemplo:

O exemplo a seguir representa que a classe CaixaMensagem é composta por uma ou mais classe Botão e por uma ou mais classe Texto. Sendo que ao instanciar a classe CaixaMensagem, as classes Botão e Texto também serão instanciadas "automaticamente"



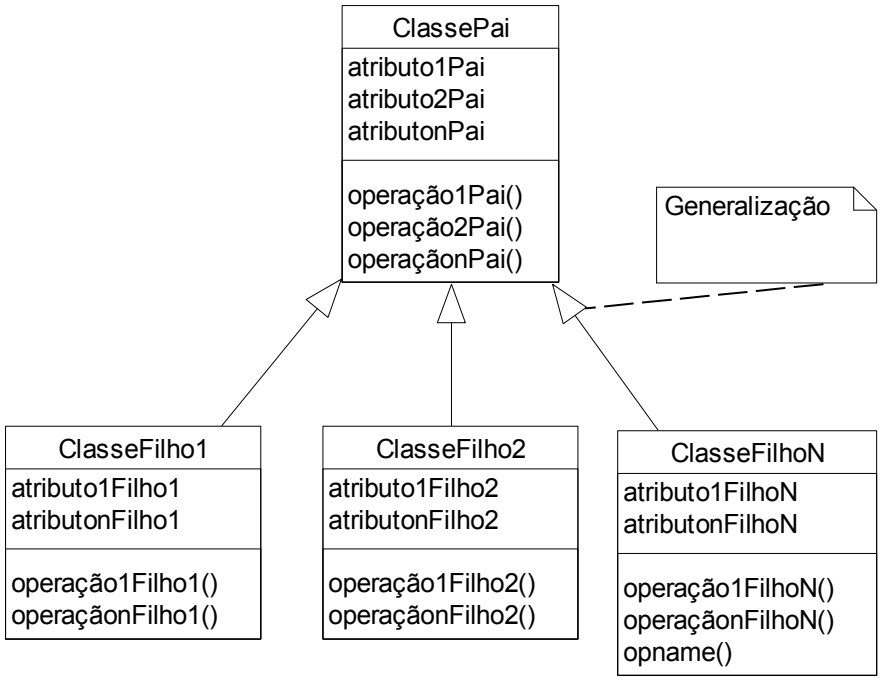
Obs: Tanto no caso da agregação, como da composição, o losango fica sempre na classe todo parte;

Todas as regras de associação (por exemplo: nome da associação, seta da associação e cardinalidade), são válidas para agregação e composição.

Pedido → itemPedido

Generalização: é um relacionamento que descreve uma relação entre grupos de "coisas" com algo em comum, que demonstra a herança de atributos e operações de uma classe pai para seus filhos.

A generalização é utilizada em classes visando a reutilização de implementação.

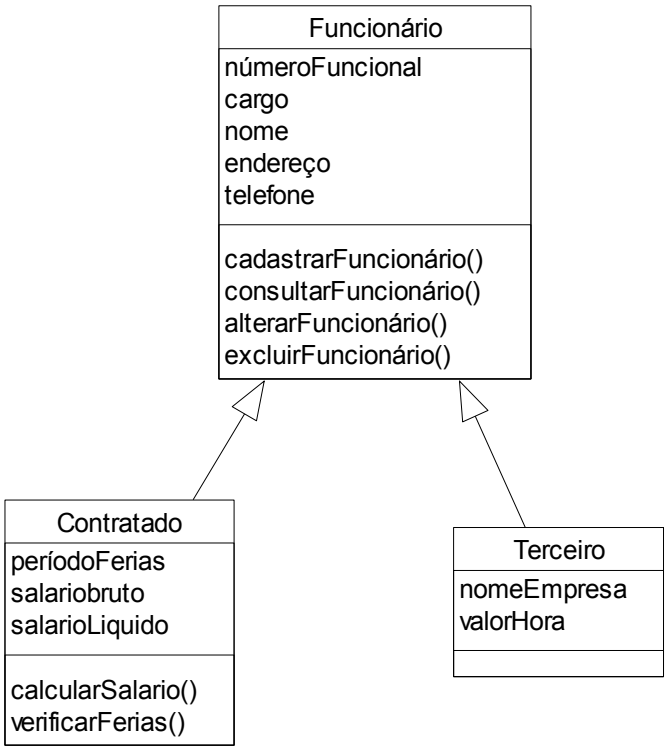


Obs: O "triângulo" (representação da generalização) deve ser representado na classe pai;
Não existe cardinalidade no relacionamento de generalização.

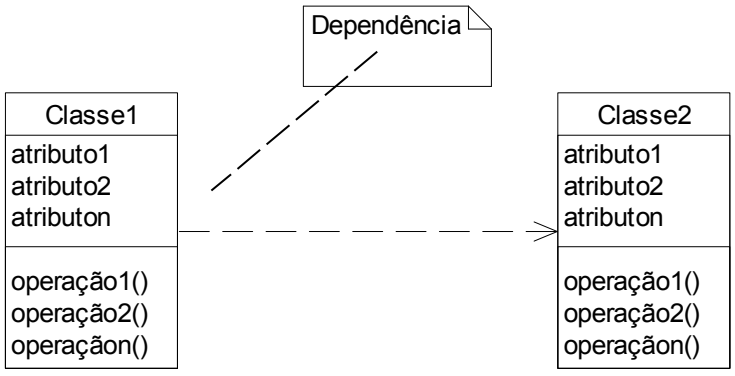
A generalização é um processo *bottom-up*, utilizada para simplificar o modelo. A partir das Classes mais especializadas, deduzem-se Classes mais genéricas contendo associações, atributos e operações que sejam comuns as Classes originais. Estes, por sua vez, são reorganizados para conter somente a parte especializada. A especialização é um processo *top-down*, que é utilizada para refinar o modelo. A partir de Classes existentes no modelo, procuram-se especializações para o mesmo, como casos particulares da classe genérica, definindo atributos e operações específicos.

Exemplo:

O exemplo a seguir representa que um funcionário pode ser um funcionário contratado ou um funcionário terceiro. Tanto o funcionário contratado, como os funcionários terceiros possuem características (atributos) e ações (operações) comuns, que são representados na classe pai Funcionário. E as características (atributos) e ações (operações) específicas ficam nas classes filhas (Contratado e Terceiro).

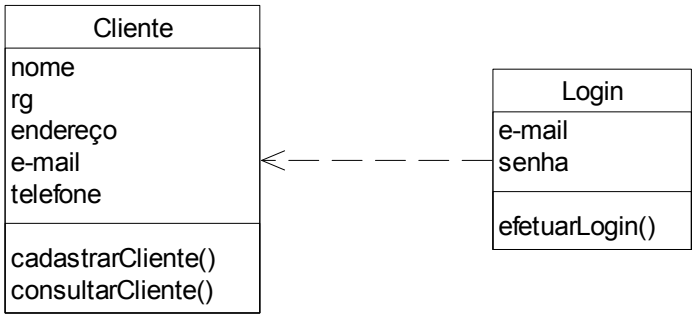


Dependência: é um relacionamento de utilização, determinando que uma classe utilize atributos e operações de outra classe, mas não necessariamente o inverso. No exemplo abaixo uma alteração na Classe2 pode afetar a Classe1, pois esta utiliza a Classe2.



Exemplo:

A classe login depende da classe cliente, pois considera-se neste exemplo que ao efetuar o login os dados do cliente devem ser exibidos.



4. Procedimento para elaboração do Diagrama de Classes de Análise

4.1. Identificação das Classes

Descrição
A primeira fase da elaboração de um modelo de Classes consiste na identificação das Classes que irão compor o sistema. A partir da descrição textual do sistema (especificação de requisitos) e/ou dos diagramas e descrições dos casos de uso, serão identificadas as Classes relevantes através de sucessivos refinamentos e detalhamentos, ou seja:

- Foco nas classes de negócio;
- Classes se originam de:
  - Especificações de Requisitos;
  - Diagramas de Casos de Uso;
  - Modelos de Processo;
  - Entrevistas;
  - Reuniões.

Estrutura da Solução

1. Identifique os substantivos na descrição do sistema (Requisitos, especificação de casos de uso, diagrama de casos de uso, etc);
2. Anote-os separadamente e classifique-os;
3. Refine os objetos identificados, avaliando os que devem permanecer no modelo;
4. Crie um esboço do diagrama com os objetos identificados;
5. Não se preocupe com atributos, operações e relacionamentos;
6. Tente refiná-los, agrupando os objetos fortemente relacionados

- **Dicas**
  - Perguntas para auxiliar na identificação de classes:
    - Existem informações que devem ser armazenadas ?
    - Existem Sistemas Externos ?
    - Existentes representações organizacionais ?
    - Os atores representam algo no negócio ?
    - Existem dispositivos utilizados pelo sistema ?
    - Existem bibliotecas, componentes ou outros elementos de software ?

- Dirija a modelagem segundo o problema a resolver.
- Escolha os nomes das Classes com critério.
- Analise cuidadosamente o texto, examinando também pronomes, para não omitir substantivos implícitos.
- Não elimine Classes a não ser que tenha absoluta certeza. É sempre possível eliminar uma Classe mais tarde.
- Planeje a localização de suas Classes no diagrama. Facilitando enxergar os relacionamentos.

4.2. Identificação de Associações entre Classes

Descrição
Após a identificação das Classes que fazem parte do sistema, é necessário relacionar as Classes de acordo com os seus papéis. Uma Classe isolada no sistema geralmente não é útil, portanto toda Classe tem algum tipo de relacionamento com outra Classe.

Identificar inicialmente as associações e agregações entre as Classes do sistema e posteriormente as generalizações/herança.

Estrutura da Solução

1. Associações
- Marque os verbos e expressões que caracterizam os substantivos identificados como objetos*;*
- Identifique as associações entre os objetos;
- Dê nomes às associações
- Identifique as associações entre as Classes. Num primeiro momento não tente distinguir as agregações entre as associações.
- Num segundo momento distinguir quais associações podem ser agregações ou composições.

2. Cardinalidades
- Estime a cardinalidade das associações.

3. Hierarquia de generalização/herança
- No diagrama de caso de uso os atores que possuem generalização/herança tendem a ser classes representadas através da generalização/herança. Procure no texto de descrição do sistema e/ou na descrição dos casos de uso, casos de Classes especializadas. Algumas Classes podem não estar muito explícitas, ou podem não parecer que são especializadas. Aproveite para examinar seu modelo novamente contra a descrição textual, olhando o sistema do ponto de vista destes agrupamentos.
- Tente definir Classes generalizadas e agrupar seus casos particulares.
- Refine o modelo, observando as dicas abaixo. Não exagere. Ao descobrir as possibilidades do mecanismo de herança, o entusiasmo é comum e tenta-se ver a generalização em todo lugar. Cuidado para não complicar demais o modelo.

Dicas
- As cardinalidades dizem respeito a cada extremidade de uma associação.

- As cardinalidades podem mudar dependendo do contexto em que a associação ocorre, do domínio do problema e das regras de negócio.

4.3. Identificação de atributos e operações

A definição dos atributos e operações na classe é baseada nas responsabilidades. Responsabilidade é um contrato de uma determinada classe. Por exemplo, uma classe Sensor é responsável por medir a temperatura e disparar um alarme, caso a temperatura alcance um determinado ponto. Nesse exemplo é fácil identificar os atributos (valor temperatura obtida, valor temperatura máximo) e operações (medir temperatura e disparar alarme).
Ao realizar a modelagem um bom ponto de partida para identificar as responsabilidades.
Técnicas, como CRC (*Class Responsability Collaboration*) e análises baseadas em casos de uso são de grande ajuda.

Estrutura da Solução para Atributos

- A partir da descrição textual do problema, tente identificar substantivos e adjetivos que representem propriedades das Classes identificadas.
- Tente descobrir mais alguns atributos através de análise e discussão. Atributos podem ser omitidos da descrição, mas o conhecimento do domínio da aplicação traz à tona muitos atributos implícitos.
- Complete o modelo de Classes com os atributos encontrados e verifique se estão na Classe correta.

Estrutura da Solução para Operações

- Identifique as possíveis operações que irão manipular os atributos identificados e qual é a classe que será responsável pela execução dessa operação.
- Incluir estas operações nas classes.

Dicas

- Atributos podem ser identificados como novas Classes, dependendo do contexto. Se um atributo tem atributos, então ele é uma Classe. Se a existência do atributo parece mais importante que o valor que ele guarda, então ele uma Classe.
- Observe o nível adequado de detalhamento. Alguns atributos são mais importantes que outros. Atributos que afetam muitas operações da aplicação são altamente relevantes. Outros atributos, que têm efeito pequeno, não necessitam ser considerados na análise.
- Cuidado com atributos necessários à implementação. Alguns atributos podem ser necessários somente para a linguagem onde será implementado o aplicativo, mas podem não ter função lógica no domínio da aplicação. Não considere estes atributos no diagrama de classes de análise.
- Verifique atributos fora de contexto. Se forem identificados grupos de atributos que não se relacionam dentro de uma Classe, examine melhor esta Classe. Pode ser interessante dividi-la em mais Classes, com os atributos mais relacionados juntos.
- Aproveite a identificação dos atributos para rever as suas Classes. Algumas Classes podem estar fora de foco, sem função definida no sistema.

Observações Gerais:

São sugeridas por Coad e Yourdon algumas características de seleção de Classes:

- A Classe será útil somente se a informação sobre ela for necessária para o funcionamento do sistema.
- A Classe em potencial deve ter um conjunto de operações identificáveis que podem mudar o valor de seus atributos de alguma maneira.
- Uma Classe com um único atributo, provavelmente, será mais bem representada como um atributo de uma outra Classe.

São sugeridas por Abbot, algumas dicas para análise das descrições gramaticais de sistemas:

Gramática	Componente do Diagrama	Exemplo
Nome Próprio	Objeto	Alice, Antonio, ...
Nome comum (substantivo)	Classe	Funcionário, sala, ...
Substantivo e Adjetivo	Atributo	Nome do aluno, número da página, ...
Verbo de ação	Operação	Criar, calcular, ...
Verbo de Existência	Generalização/herança	É um tipo de, e um de, pode ser, ...
Verbo de Posse	Agregação e Composição	Consiste de, faz parte de, ...

Notação Complementar – Interface

Em muitos casos é necessário integrar uma ou mais classes com outros sistemas de software ou outros componentes de software. Na UML, utiliza-se o conceito de interface para integrar sistemas ou parte de sistemas. Uma interface é uma coleção de operações para especificar o serviço de uma classe. Declarando a interface, pode-se estabelecer o comportamento desejado de uma abstração independente da implementação (UML – Guia do Usuário).

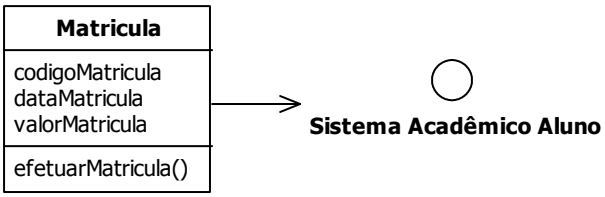
A interface pode ser representada por uma representação gráfica, apresentada a seguir ou utilizando o estereótipo <<interface>>. A interface no diagrama de classes de análise permite visualizar de forma ampla e abstrata a integração de uma classe com um sistema externo ou um componente externo, sem se preocupar com qualquer implementação.

Notação gráfica de interface:

Exemplo de Interface:

Nome Interface

Representação gráfica:



Representação com estereótipo: