USP - Universidade de São Paulo

IME - Instituto de Matemática e Estatística

MAC0332

Engenharia de Software

Análise Orientada a Objetos

Marco Aurélio Gerosa

gerosa@ime.usp.br

O que é um paradigma?

- Dicionário Aurélio:
  - 1. Modelo, padrão, estalão
  - 2. Termo com o qual Thomas Kuhn (v. kuhniano) designou as realizações científicas (p. ex., a dinâmica de Newton ou a química de Lavoisier) que geram modelos que, por período mais ou menos longo e de modo mais ou menos explícito, orientam o desenvolvimento posterior das pesquisas exclusivamente na busca da solução para os problemas por elas suscitados.
- Wikipedia:
  - Paradigma (do grego Parádeigma) literalmente modelo, é a representação de um padrão a ser seguido. É um pressuposto filosófico, matriz, ou seja, uma teoria, um conhecimento que origina o estudo de um campo científico; uma realização científica com métodos e valores que são concebidos como modelo; uma referência inicial como base de modelo para estudos e pesquisas.

Paradigmas

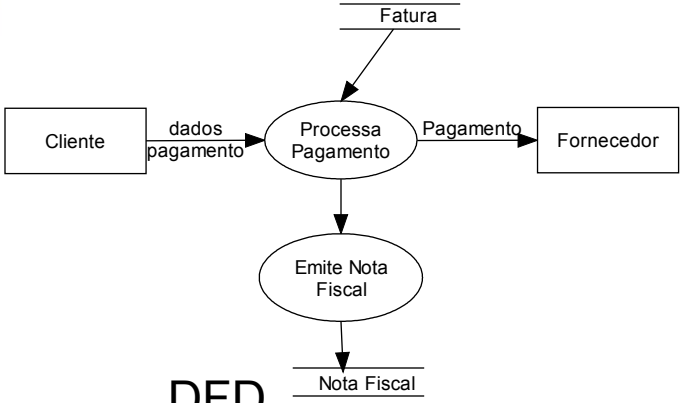
- Quais as vantagens?
- Quais os problemas? (exemplo da histórica dos macacos e a escada)

Paradigmas de desenv. de software

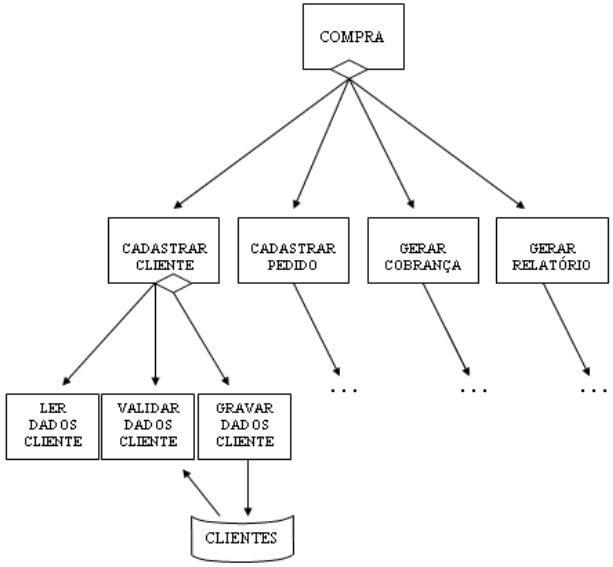
- Análise Estruturada
- Orientação a objetos
- Programação orientada a aspectos
- Desenvolvimento baseado em componentes
- Etc.

Análise estruturada

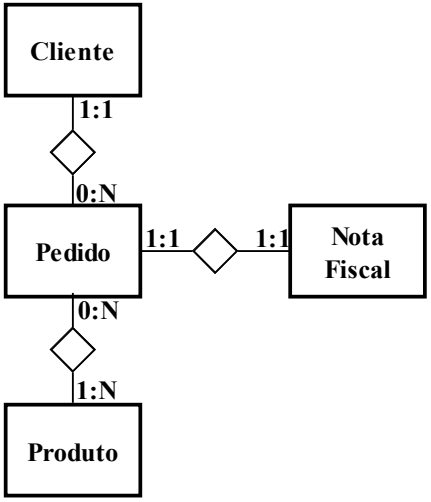
- Análise dos processos (funcionalidade) e dos dados. A separação tem origem na arquitetura de von Neumann.



DFD

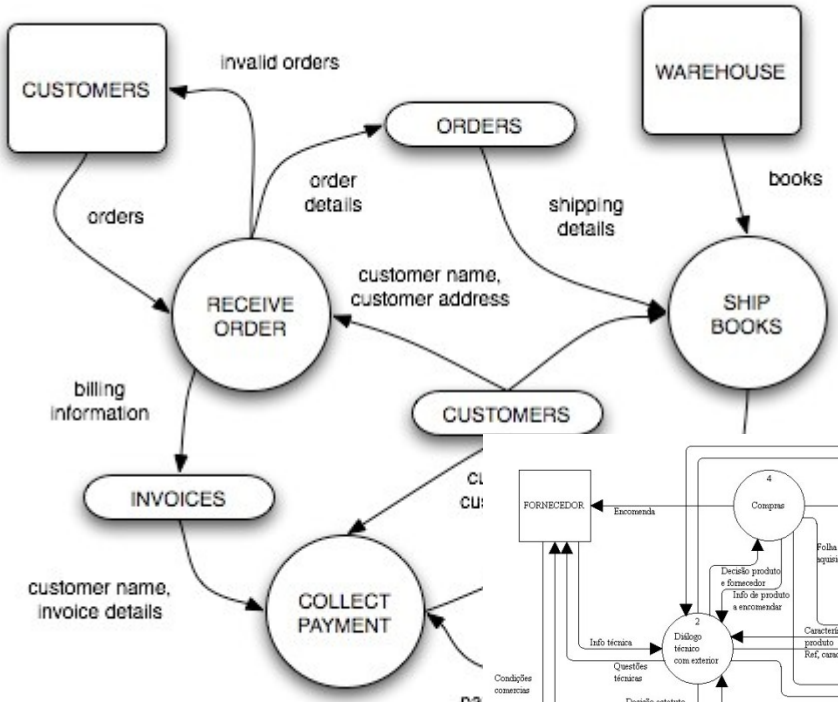


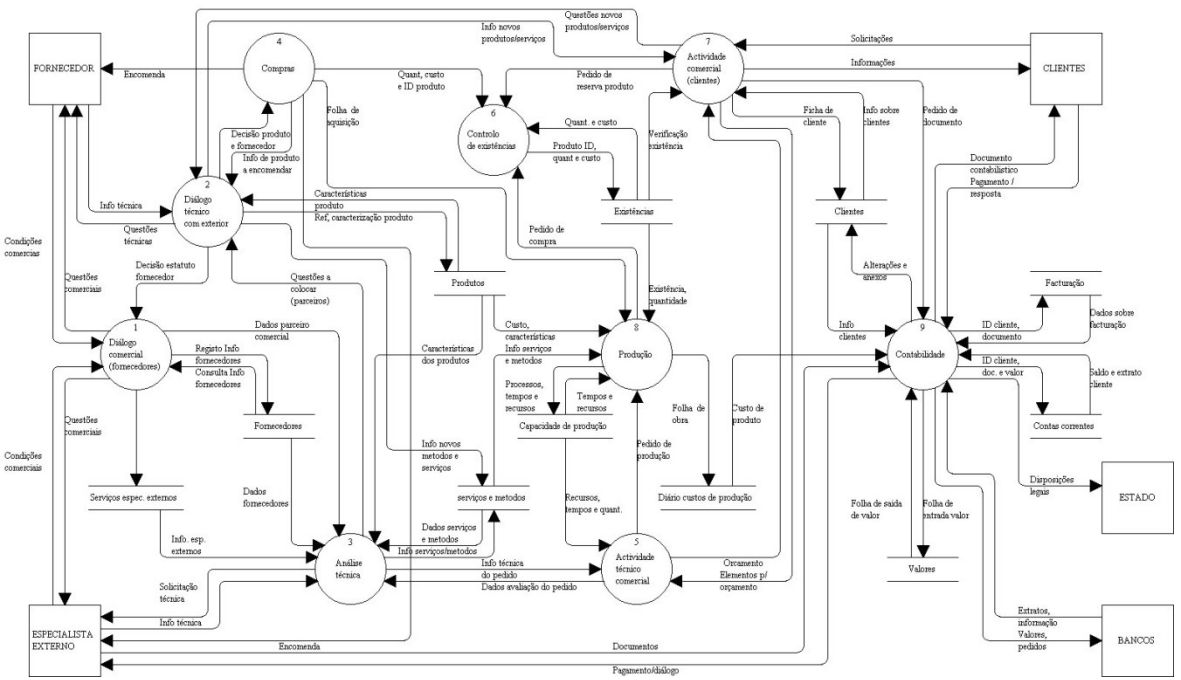
DEM



DER

DFD


CUSTOMERS
invalid orders
ORDERS
WAREHOUSE
orders
order details
shipping details
books
RECEIVE ORDER
customer name, customer address
SHIP BOOKS
billing information
CUSTOMERS
INVOICES
customer name, invoice details
COLLECT PAYMENT

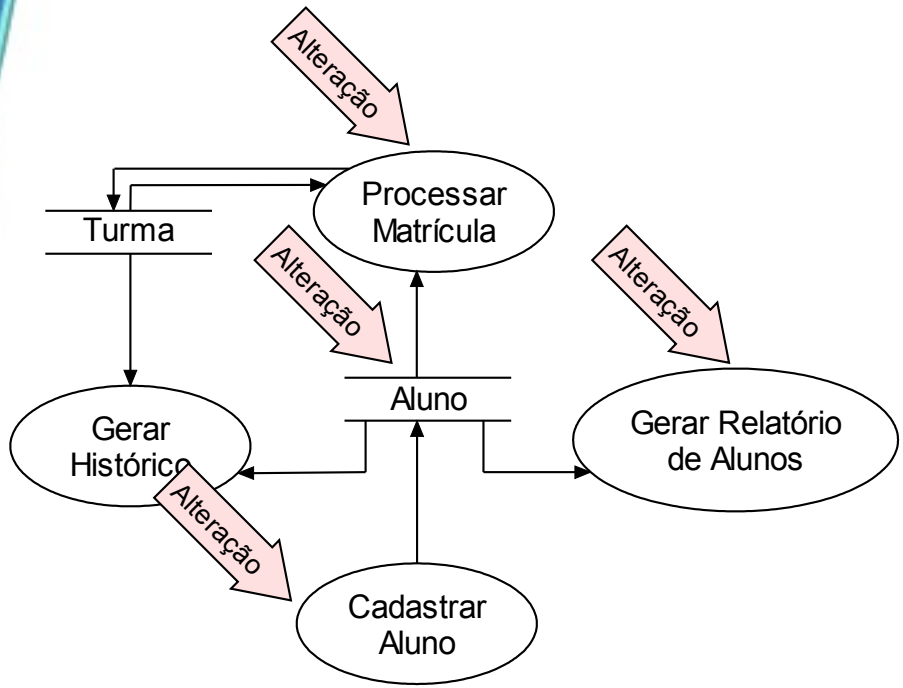

FORNECEDOR
Encomenda
Compras
Quant, custo e ID produto
Folha de aquisição
Decisão produto e fornecedor
Info de produto a encomendar
Info novos produtos/serviços
Questões novos produtos/serviços
Actividade comercial (clientes)
Solicitações
Informações
CLIENTES
Pedido de reserva produto
Controlo de existências
Quant. e custo
Produto ID, quant e custo
Verificação existência
Ficha de cliente
Info sobre clientes
Pedido de documento
Documento contabilístico Pagamento / resposta
Diálogo técnico com exterior
Info técnica
Questões técnicas
Características produto
Ref, caracterização produto
Existências
Clientes
Condições comerciais
Questões comerciais
Decisão estatuto fornecedor
Questões a colocar (parceiros)
Pedido de compra
Produtos
Existência, quantidade
Alterações e anexos
Facturação
Dados sobre facturação
Diálogo comercial (fornecedores)
Registo Info fornecedores
Consulta Info fornecedores
Dados parceiro comercial
Custo, características
Info serviços e metodos
Produção
Características dos produtos
Info clientes
Contabilidade
ID cliente, documento
ID cliente, doc. e valor
Saldo e extrato cliente
Processos, tempos e recursos
Tempos e recursos
Capacidade de produção
Folha de obra
Custo de produto
Contas correntes
Fornecedores
Pedido de produção
Info novos metodos e serviços
Serviços espec. externos
Dados fornecedores
serviços e metodos
Recursos, tempos e quant.
Diário custos de produção
Disposições legais
ESTADO
Folha de saída de valor
Folha de entrada valor
Valores
Info. esp. externos
Dados serviços e metodos
Info serviços/metodos
Análise técnica
Actividade técnico comercial
Info técnica do pedido
Dados avaliação do pedido
Orçamento
Elementos p/ orçamento
Solicitação técnica
Info técnica
ESPECIALISTA EXTERNO
Encomenda
Documentos
Pagamento/diálogo
Extratos, informação Valores, pedidos
BANCOS

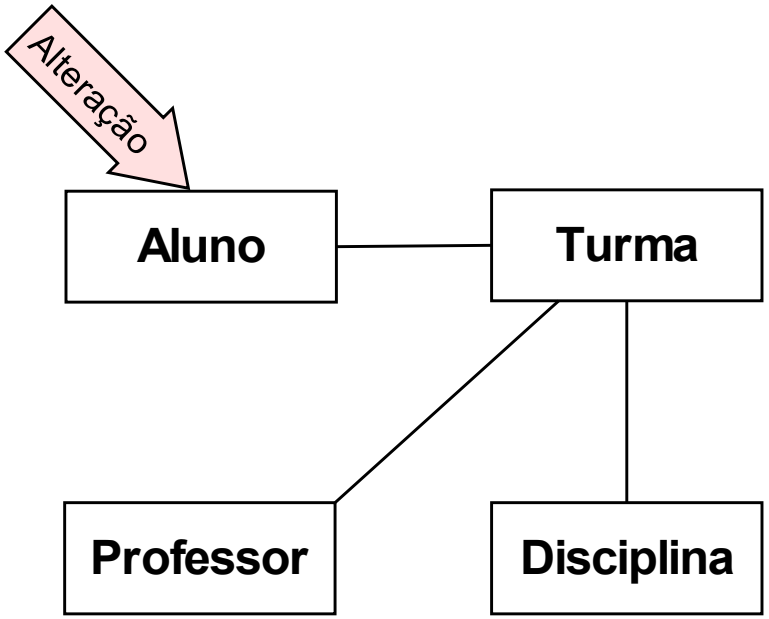
Problemas da análise estruturada

- Dificuldade de manter a documentação
- Alterações constantes nas funcionalidades impactam a arquitetura da aplicação

Orientação a objetos

- Objeto = dados + funções
- Mais estabilidade arquitetural, maior encapsulamento





Benefícios da orientação a objetos

- Conceitos do mundo real são utilizados em diferentes níveis de abstração

Mundo real:
Conceito

Análise:
Classe conceitual

Livro
título numPaginas
abrir() fechar()

Projeto:
Classe de software

Livro
- titulo : String - numPaginas : int
+ abrir() + fechar()

Implementação:
Classe em uma linguagem de programação

```
class Livro {
    private String titulo;
    private int numPaginas;
    public void abrir () {
        // código referente a operação
    }
    public void fechar () {
        // código referente a operação
    }
}
```

- Facilita o ciclo de desenvolvimento iterativo
- Menor gap semântico (distância entre o domínio do problema – mundo real – seu modelo) => melhor manutenção e legibilidade

Benefícios da orientação a objetos

- Estruturação mais estável e código menos acoplado, o que diminui os custos de manutenção;
- os modelos são mais próximos da realidade, o que diminui o gap semântico e torna os modelos mais fáceis de entender;
- facilita a comunicação entre os stakeholders, que passam a trabalhar e pensar em termos do domínio da aplicação e participar mais diretamente da análise;
- favorece o reuso, já que podem ser reaproveitados componentes genéricos, estáveis e independentes;
- adequa-se melhor ao processo de desenvolvimento iterativo, visto que as atividades do ciclo de vida compartilham vocabulário, notação e estratégias comuns.

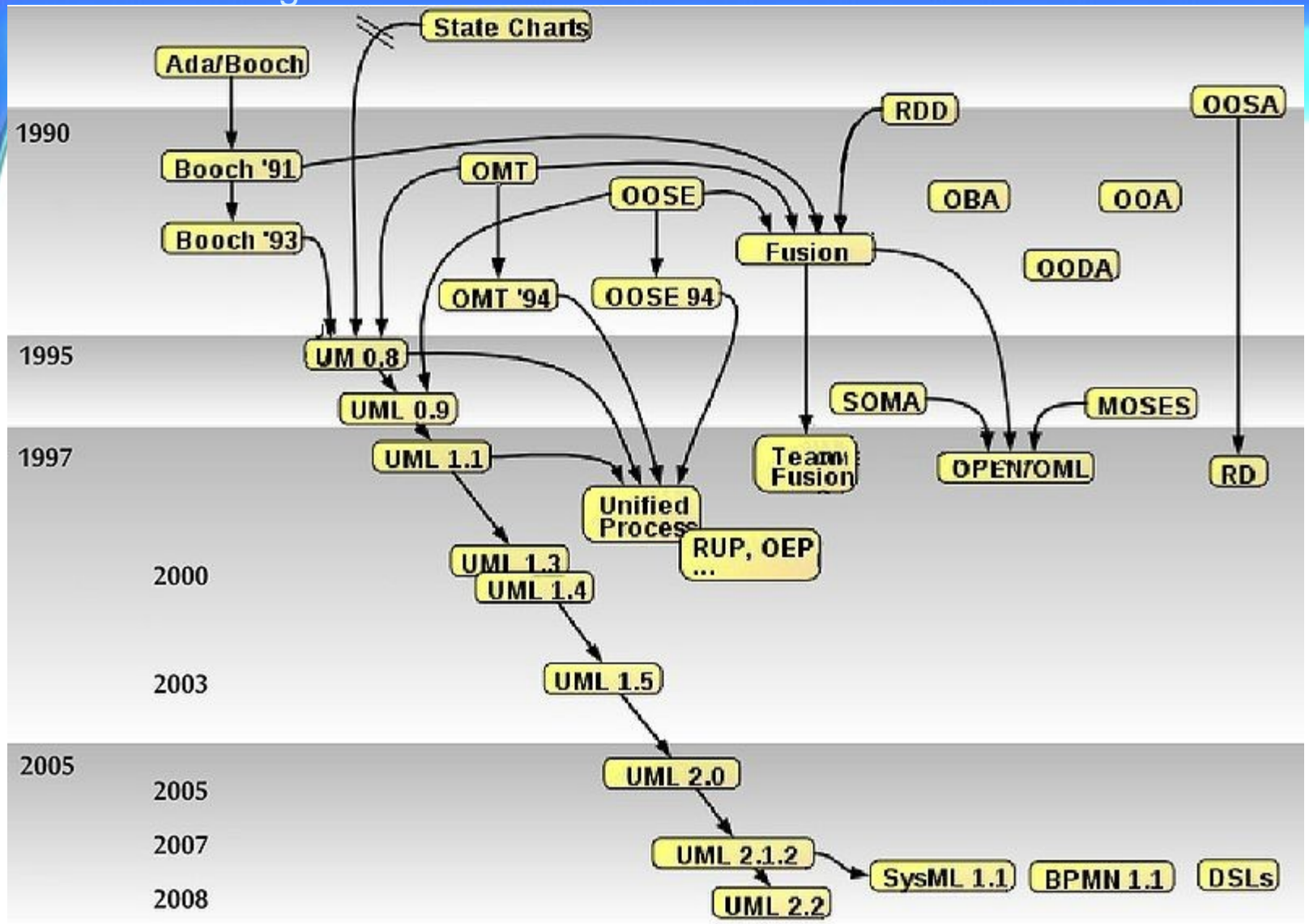
- Silver bullet do desenvolvimento de software?

Análise

- Análise x projeto
- Modelagem conceitual
- Modelo
- Abstração
- Encapsulamento
- Identificação de classes, relacionamentos, atributos e operações
- Técnicas de modelagem:
  - Identificando substantivos, brainstorm, cartões CRC
- Diagrama de classes:
  - Nome associação, direção de leitura, cardinalidade (ou multiplicidade), papel, navegabilidade, restrições, observações, estereótipo
  - Classe associativa

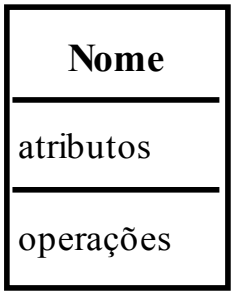
UML

- 1994 – Grady Booch e Jim Rumbaugh unificam os métodos Booch e OMT (Object Modeling Technique)
- 1995 – Versão 0.8 UML (Unified Modeling Language)
- 1996 – Ivar Jacobson, criador do método Objectory, junta-se ao grupo.
- Os três fundam a Rational Software e lançam a versão 0.9 da linguagem
- Em 1997 lançaram a linguagem como proposta de padronização à OMG (Object Management Group), que homologou-a como padrão.
- Grupo RTF (Revision Task Force) para revisão da linguagem
- 1998 – versões 1.2 e 1.3
- 2001 – versão 1.4
- 2005 – versão 2.0
- (...)
- 2010 – versão 2.3

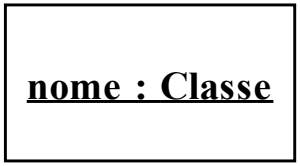


http://en.wikipedia.org/wiki/File:OO-historie.jpg

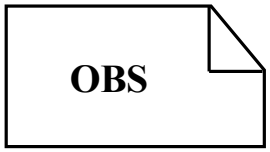
Elementos da UML



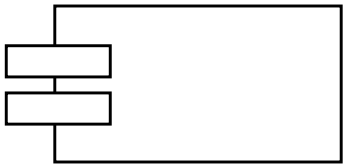
Classe (*class*)



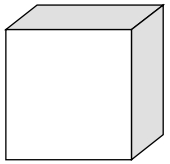
objeto (*object*)



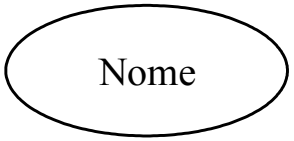
Nota (*note*)



componente (*component*)



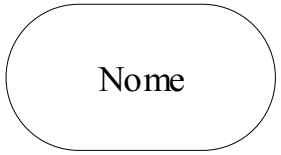
Nó (*node*)



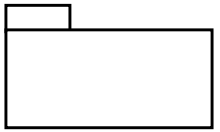
Caso de uso (*use case*)

ator (*actor*)

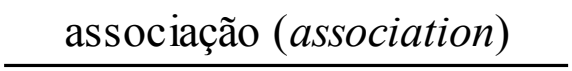
estado (*state*)

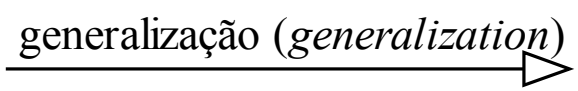


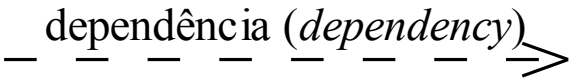
atividade (*activity*)

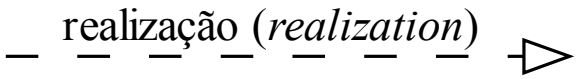


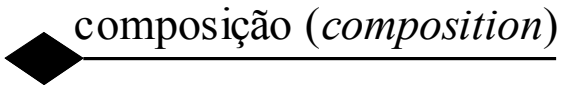
pacote (*package*)

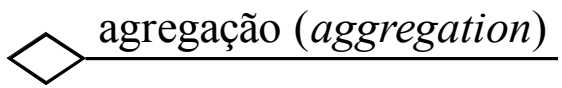




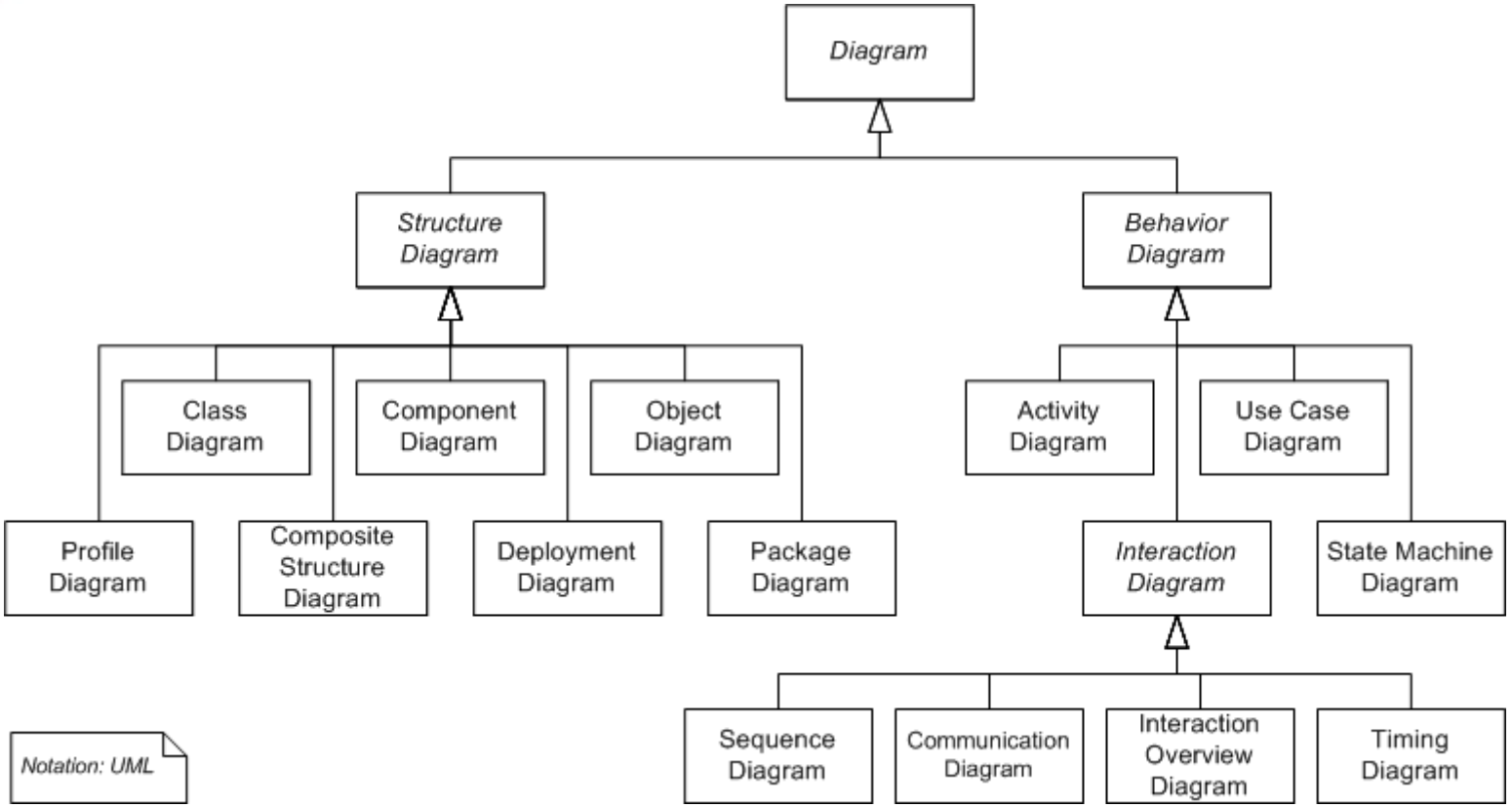








Diagramas UML



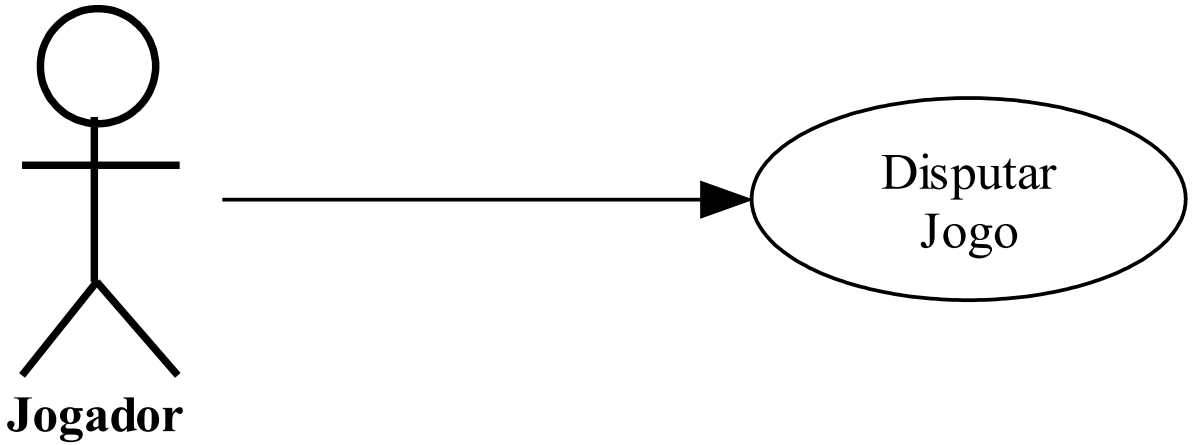
http://en.wikipedia.org/wiki/File:Uml_diagram2.png

Análise x Projeto

- Análise = fazer a coisa certa
  - Análise de requisitos
  - Análise do domínio
- Projeto = fazer certo a coisa
  - Arquitetura
  - Design
  - Infra-estrutura
- Notação x processo
- *Na preparação para uma batalha os planos são inúteis, porém o planejamento é indispensável.*
  General Eisenhower

Exemplo

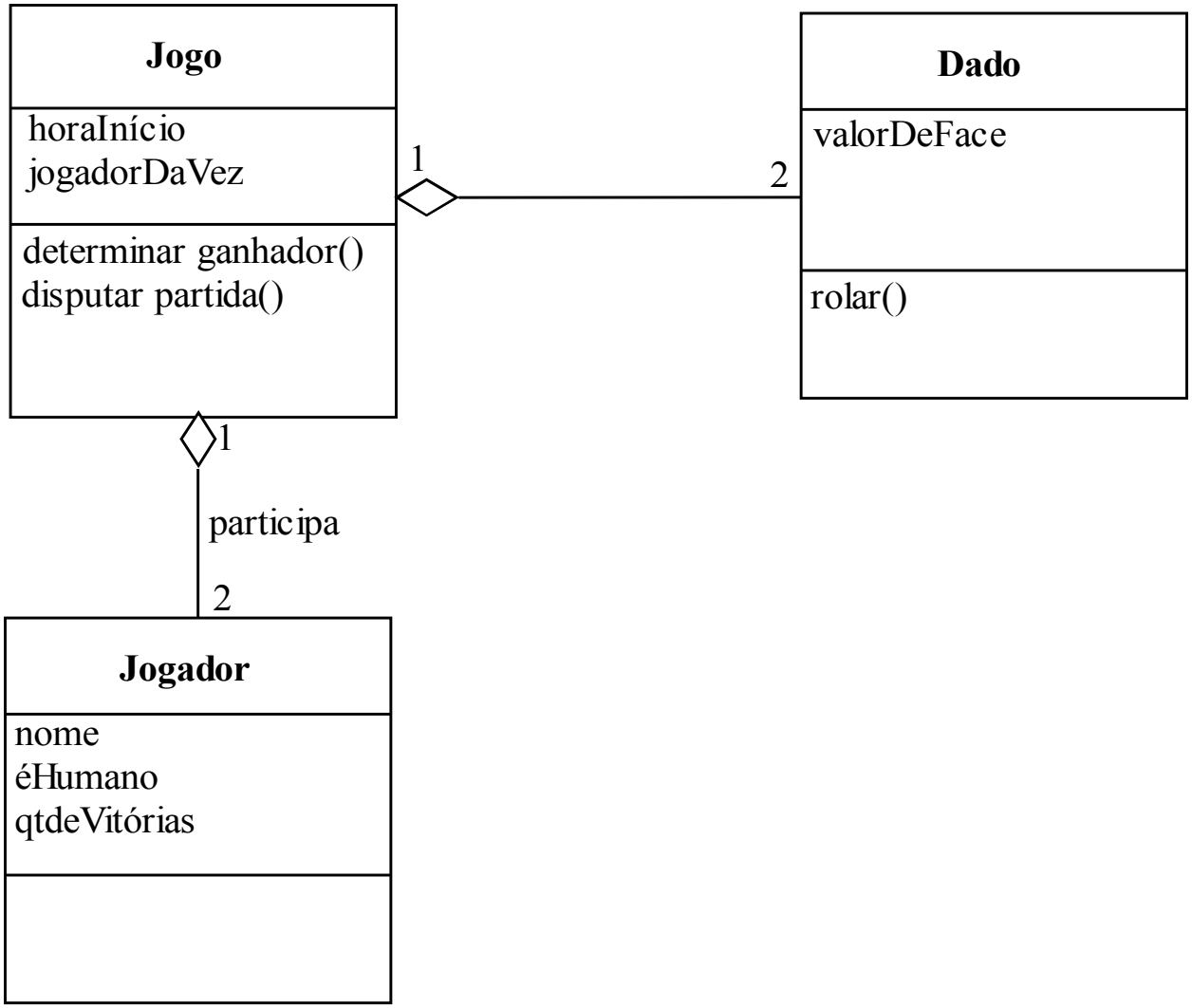
- *Descrição do problema*
- O sistema deverá simular um jogo de dados entre um jogador e o computador. Cada um joga dois dados e quem somar mais pontos ganha a partida. Ganha o jogo quem ganhar três partidas.



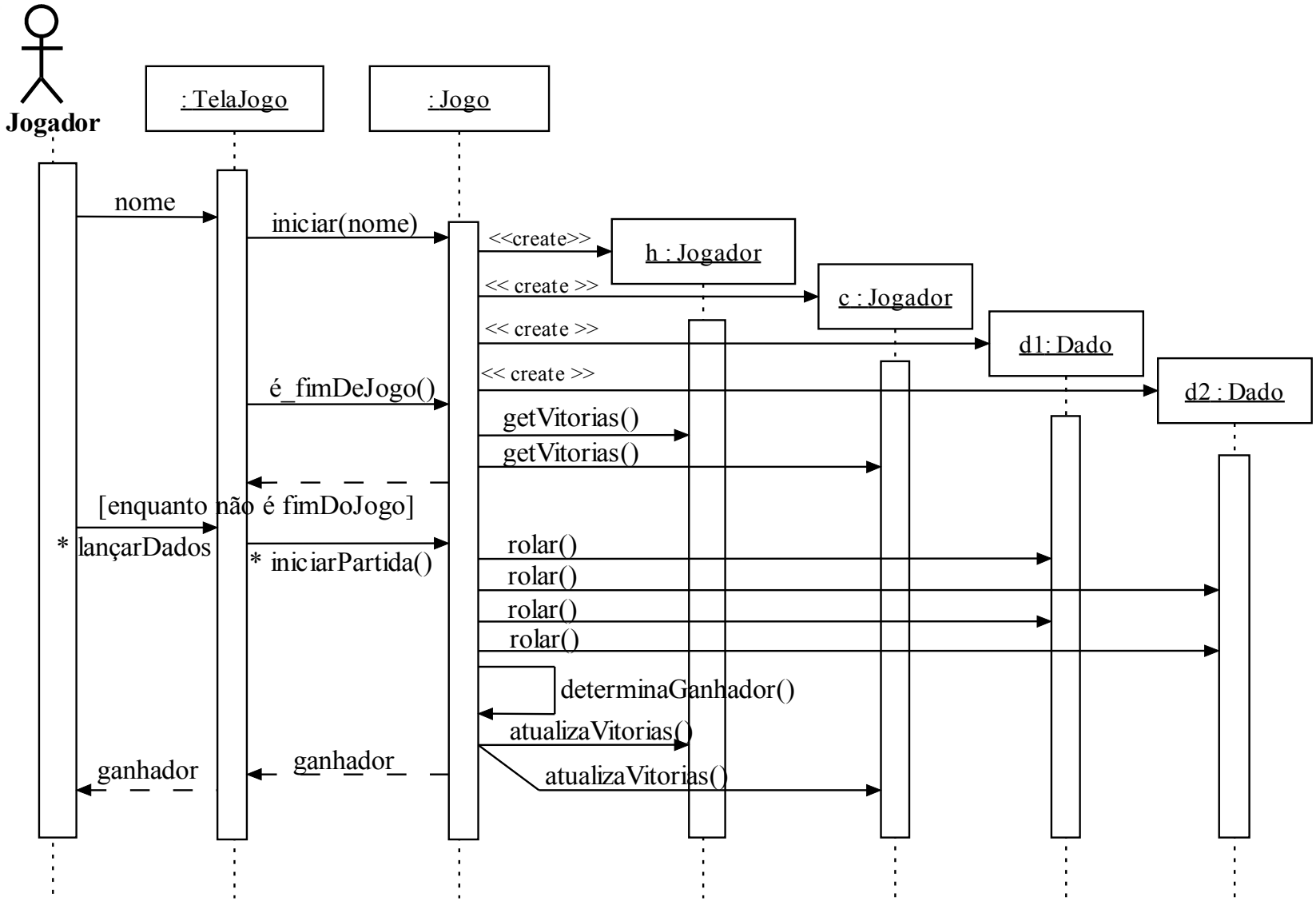
Descrição do caso de uso

- **Caso de Uso:** Disputar Jogo
- **Ator**: Jogador
- **Descrição**: O jogador disputa o jogo com o computador.
- **Fluxo principal**
- 1. O jogador informa seu nome.
- 2. O sistema registra o nome e dá início ao jogo.
- 3. O jogador informa que quer lançar os dados.
- 4. O sistema sorteia o valor de face dos dados do jogador e exibe o resultado.
- 5. O sistema sorteia o valor de face dos dados para o computador e exibe o resultado.
- 6. O sistema verifica quem obteve a maior soma de pontos e exibe o ganhador da partida.
- 7. Se nem o jogador e nem o computador obtiveram três vitórias, volta para o passo 3.
- 8. O sistema exibe o ganhador do jogo.
- **Fluxo alternativo**
- *a. A qualquer momento, o usuário decide sair.
- 1. O sistema finaliza o jogo.
- 3a. O jogador demora mais do que 3 minutos para lançar os dados
- 1. O sistema finaliza o jogo

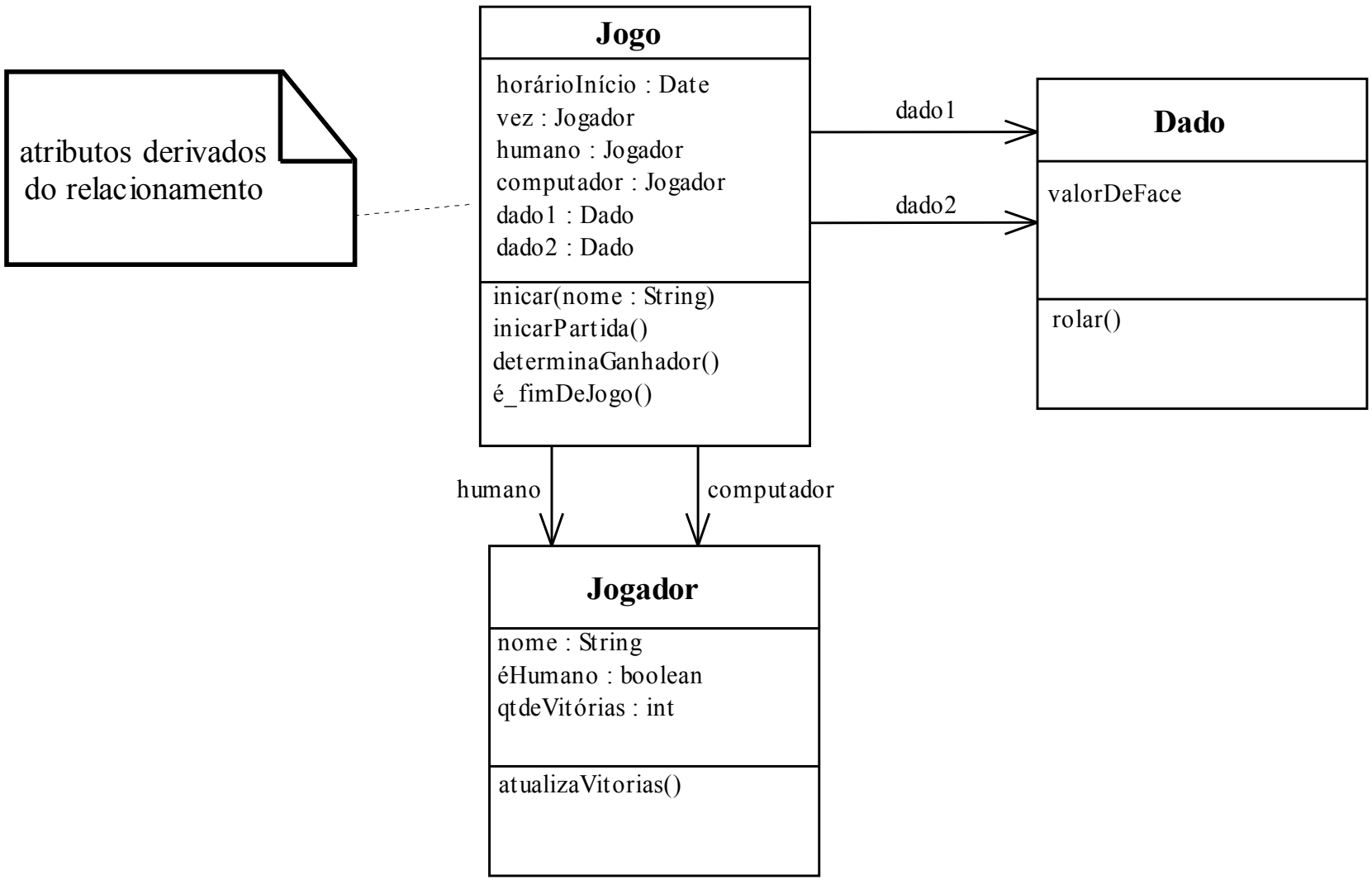
Análise do domínio



Projeto dinâmico

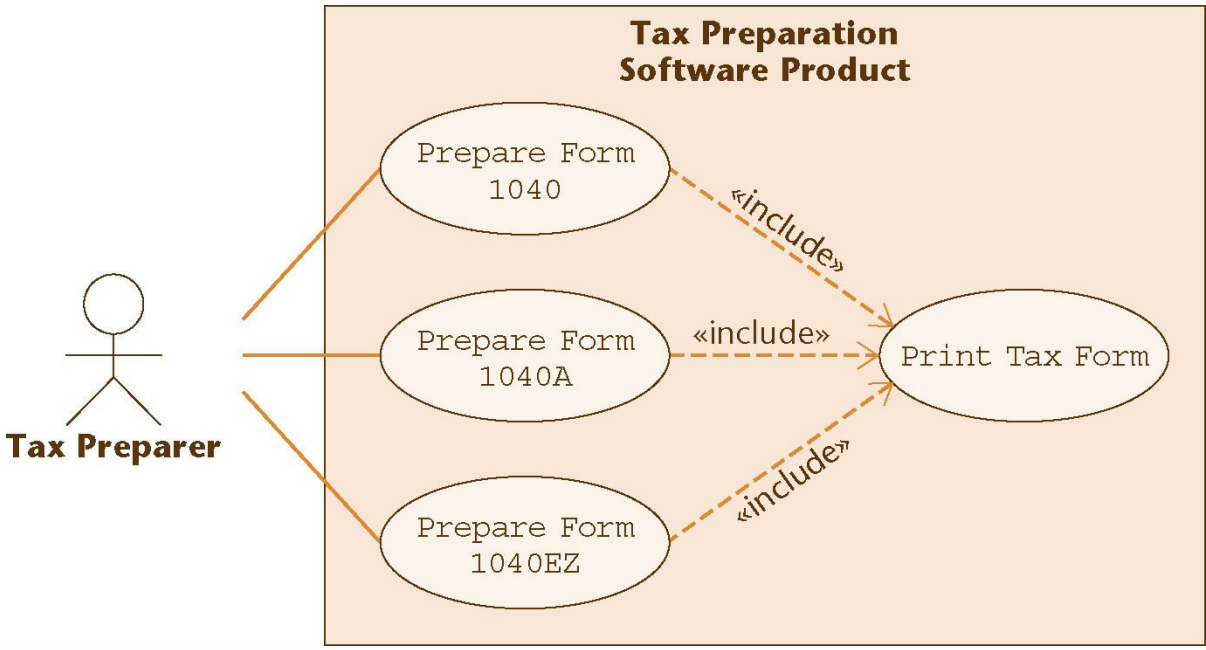


Projeto estático

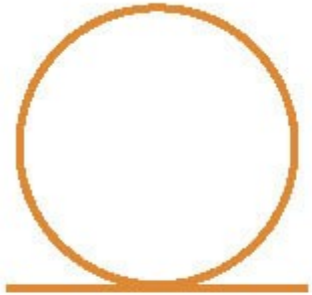


Visibilidade e estereótipo

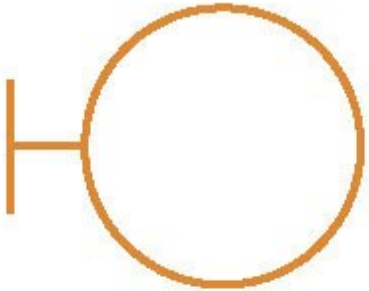
Bank Account
– accountBalance
+ deposit ( ) + withdraw ( )



Tipos de classes



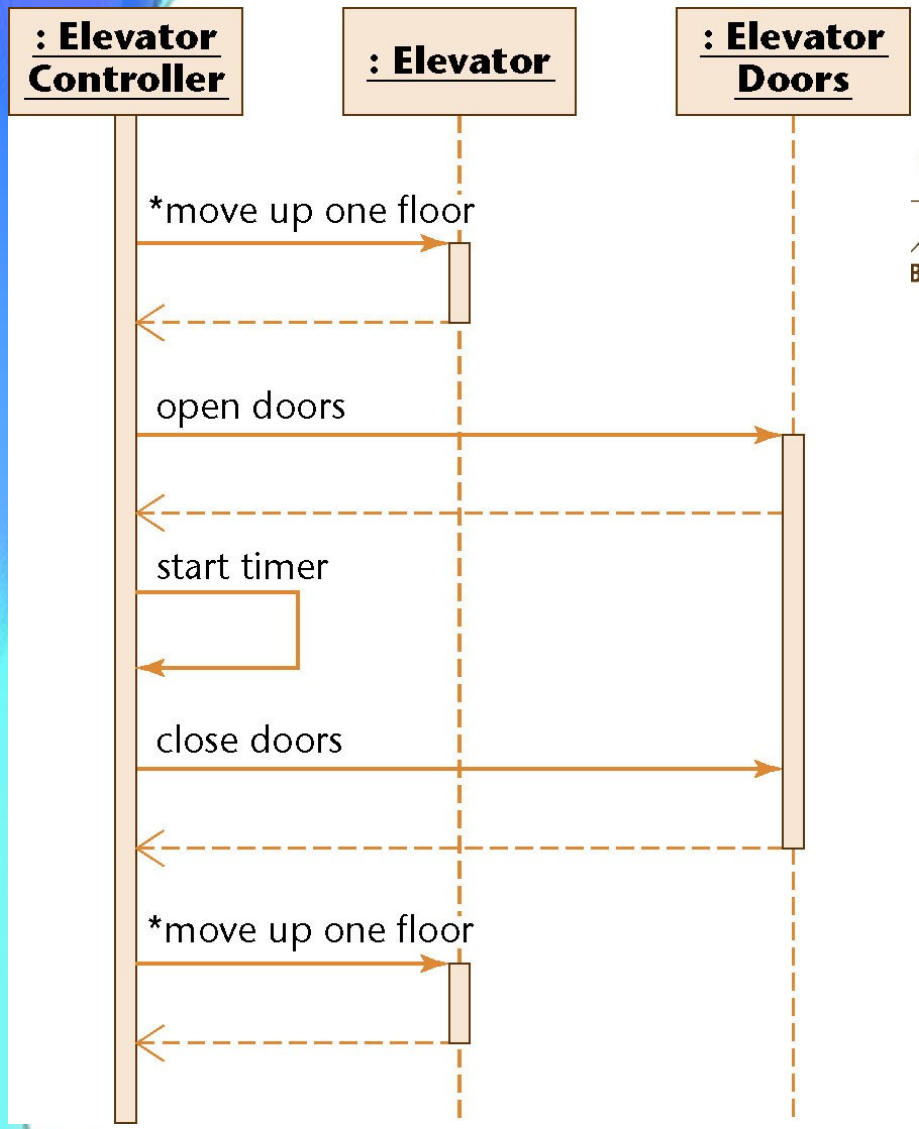
Entity Class



Boundary Class

Control Class

Diagrama de Sequência



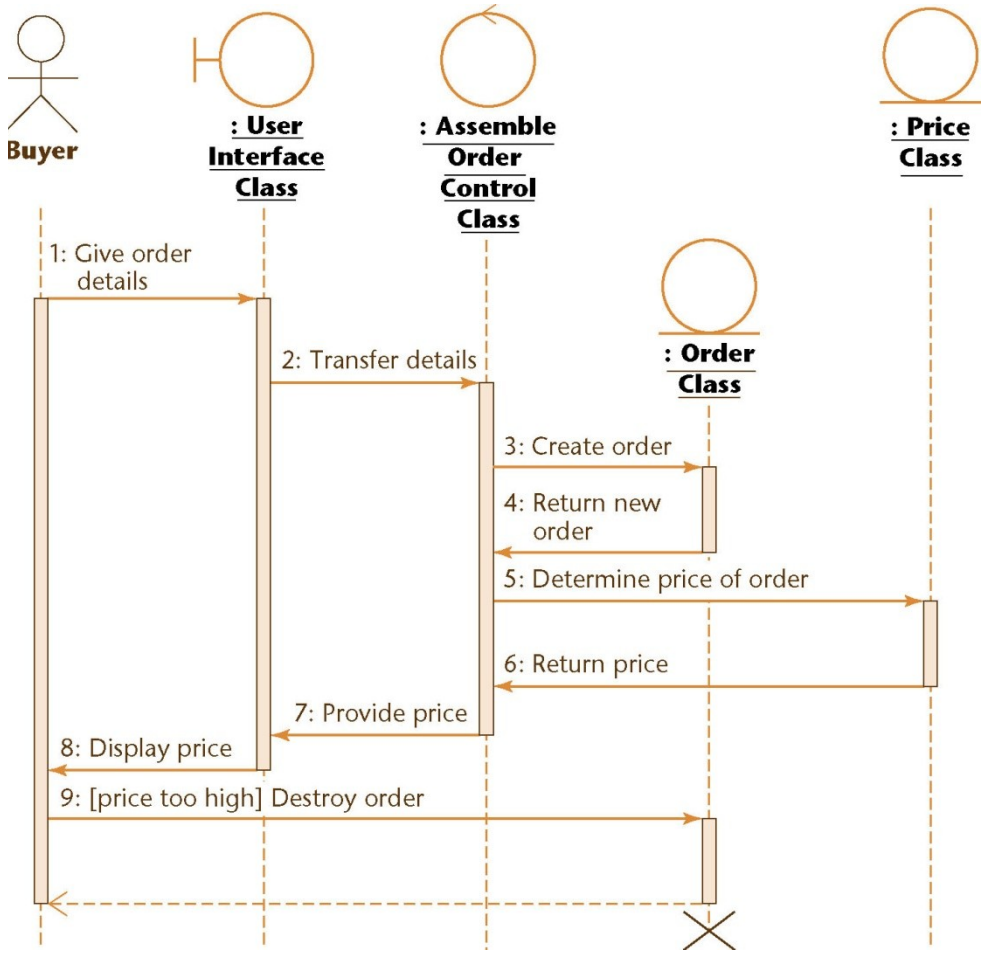


Diagrama de colaboração

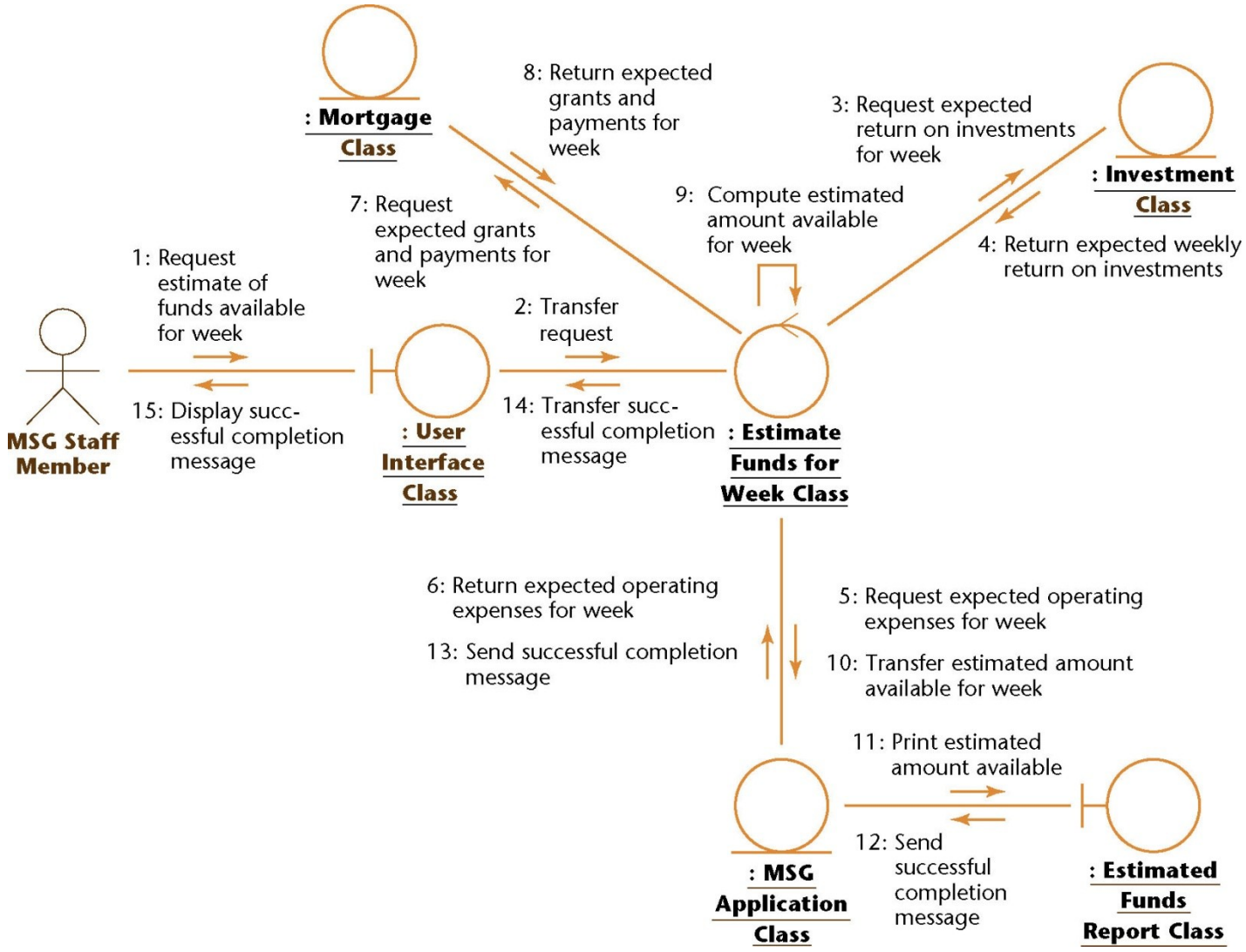


Diagrama de Sequência

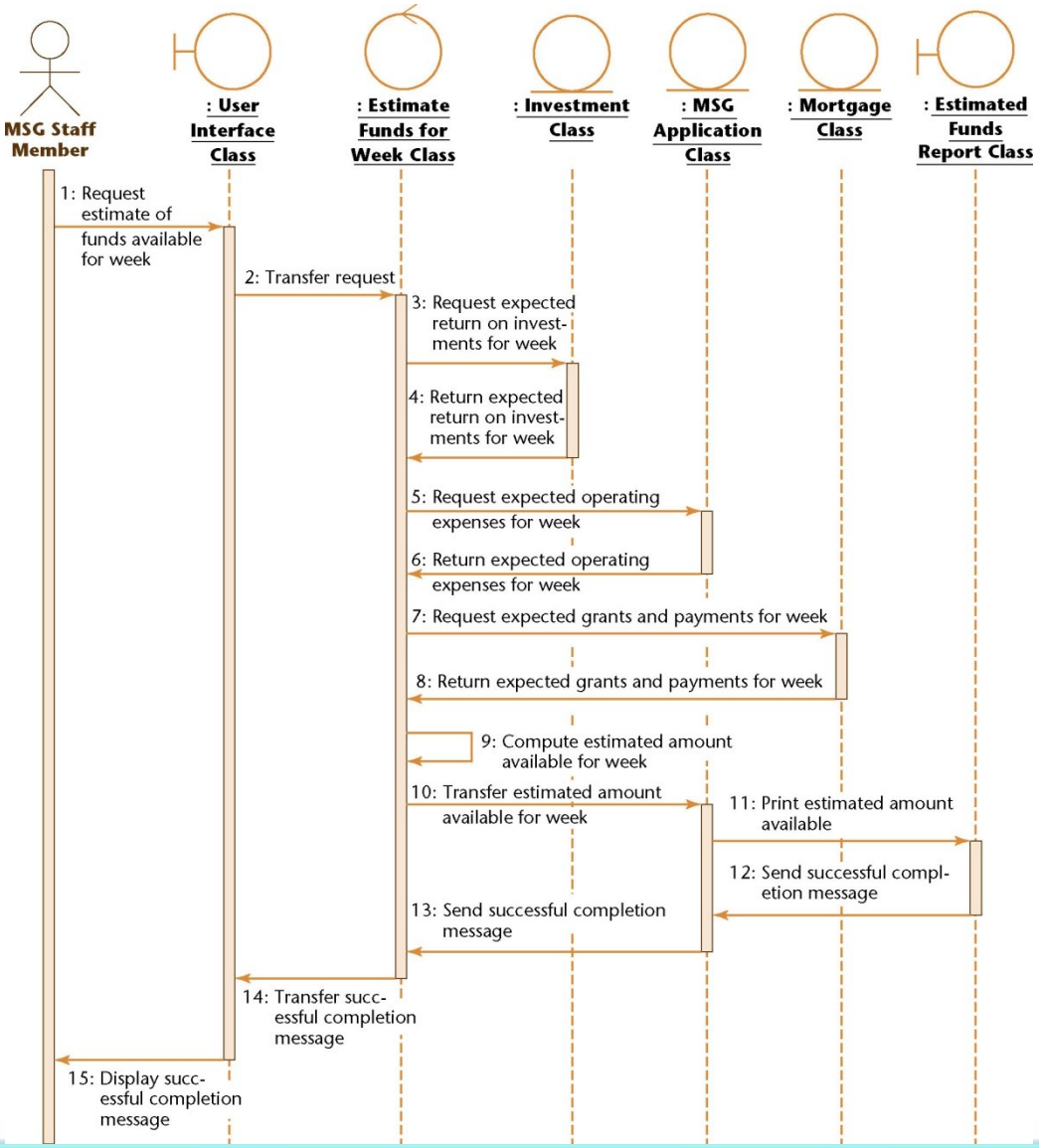
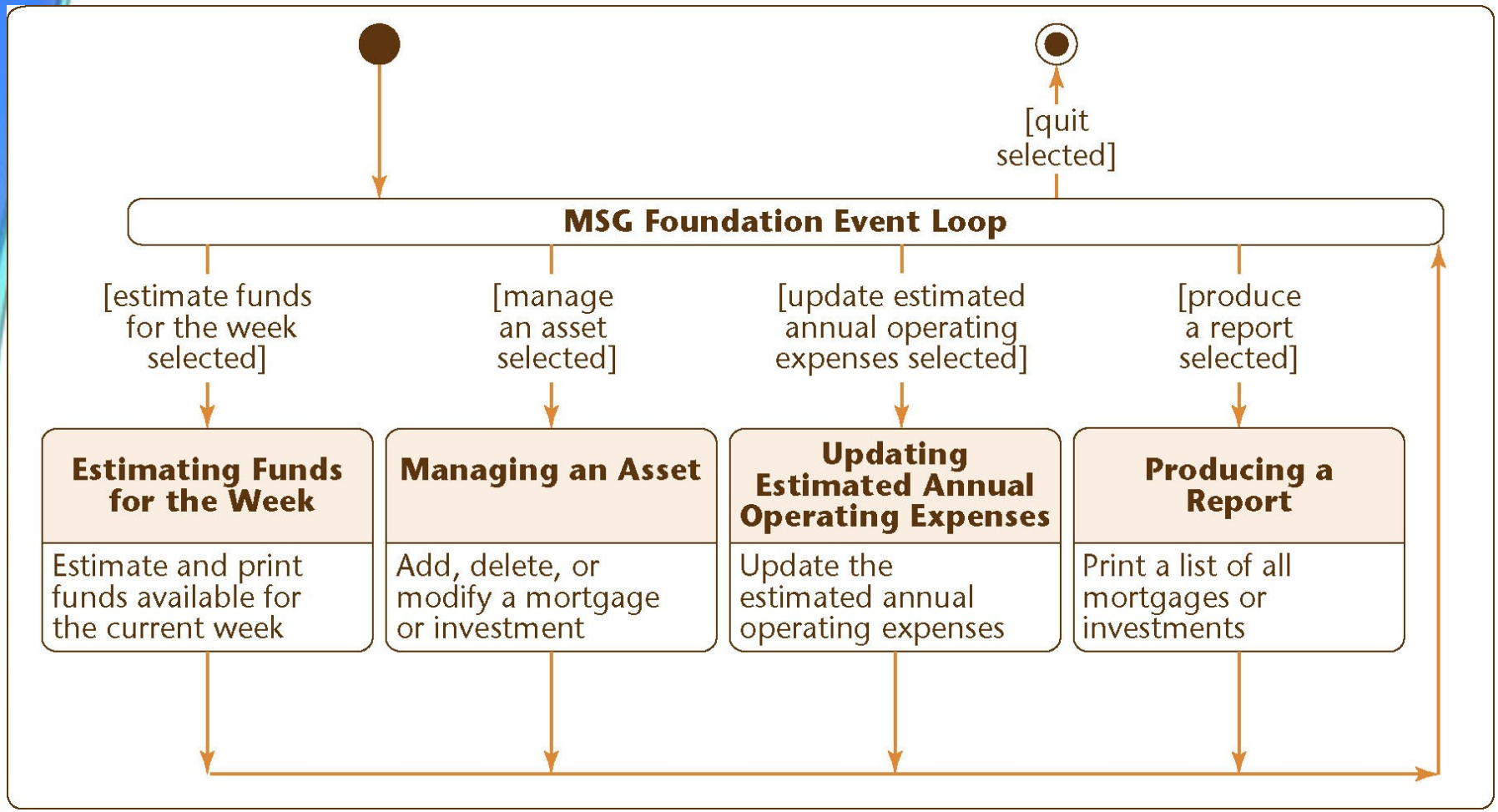


Diagrama de estados



Exemplo prático

A product is to be installed to control n elevators in a building with m floors. The problem concerns the logic required to move elevators between floors according to the following constraints:

1. Each elevator has a set of m buttons, one for each floor. These illuminate when pressed and cause the elevator to visit the corresponding floor. The illumination is canceled when the corresponding floor is visited by the elevator

2. Each floor, except the first and the top floor, has two buttons, one to request an up-elevator, one to request a down-elevator. These buttons illuminate when pressed. The illumination is canceled when an elevator visits the floor, then moves in the desired direction

3. If an elevator has no requests, it remains at its current floor with its doors closed

Use Case

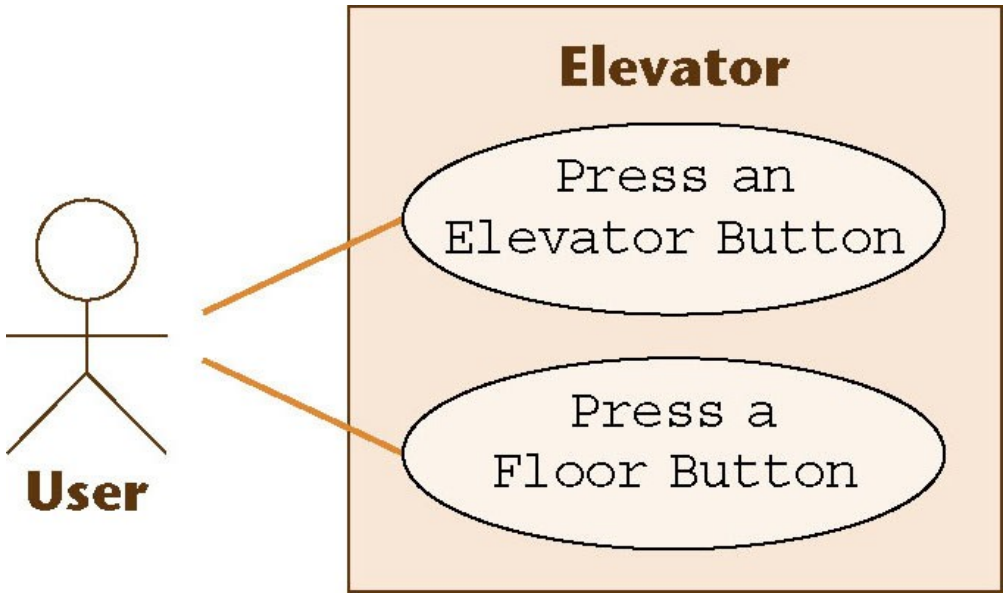

Elevator
Press an Elevator Button
Press a Floor Button
User

Diagrama de atividades

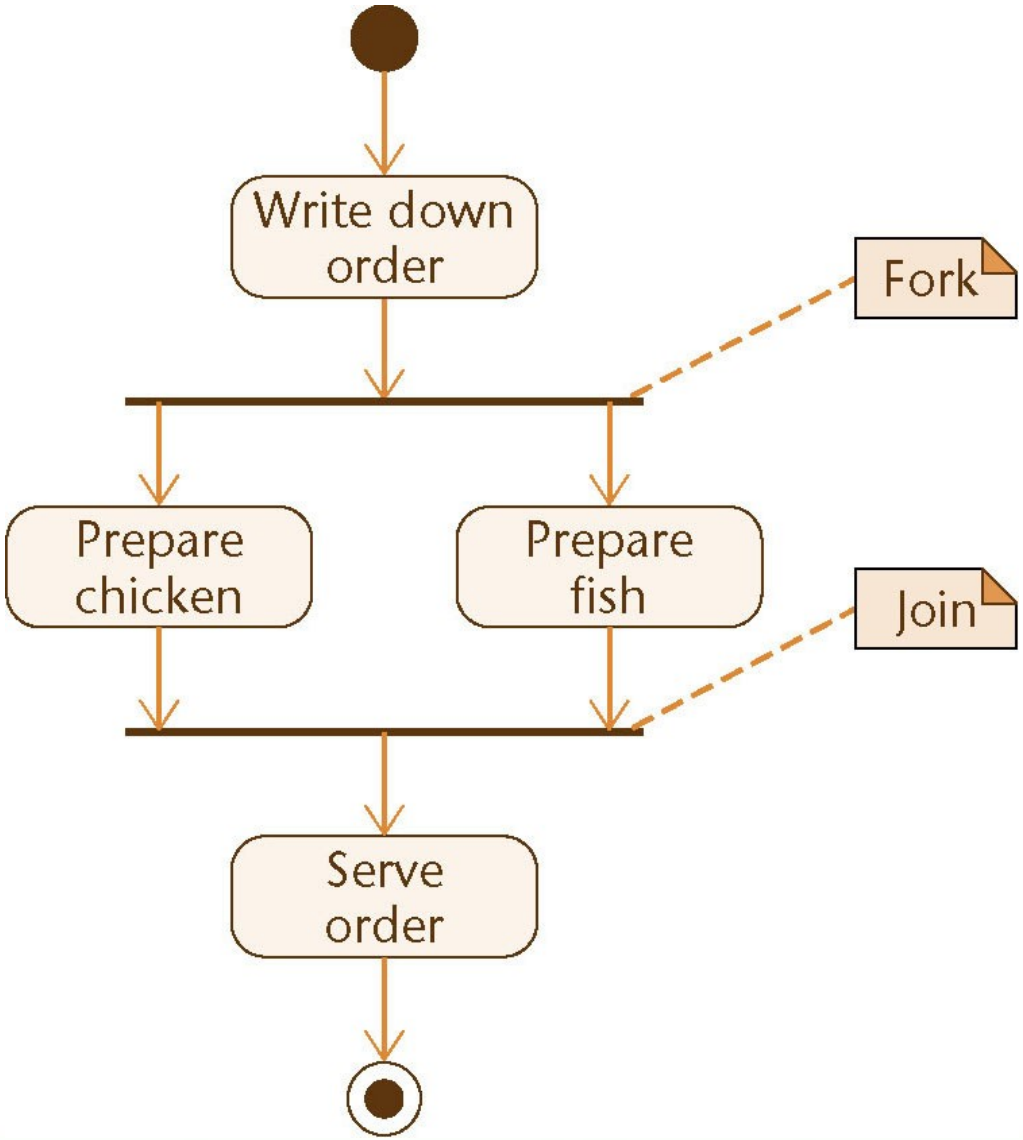
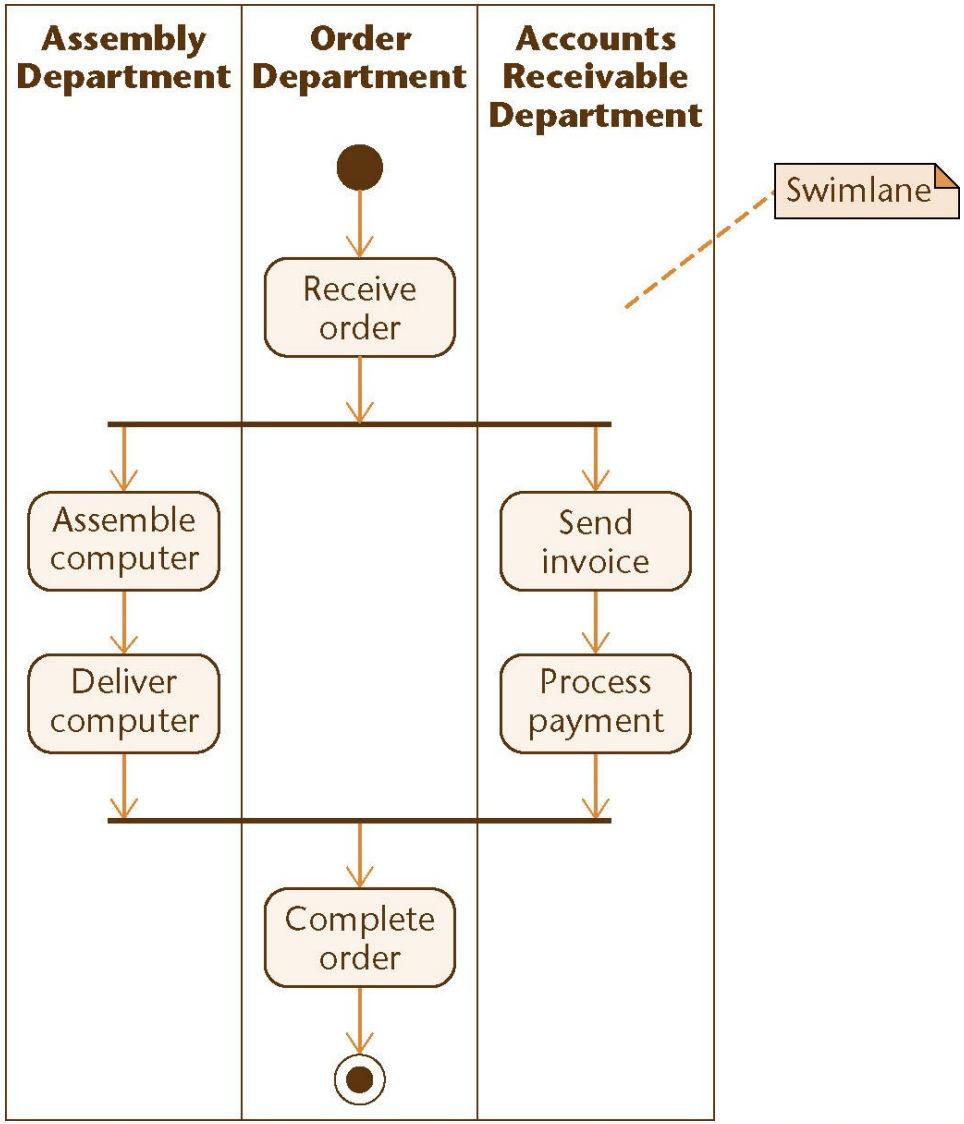
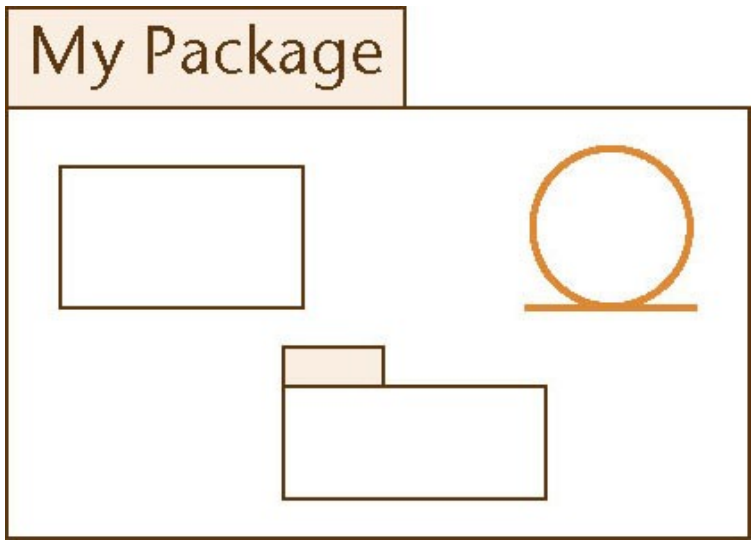


Diagrama de atividades com raias



Pacotes



Cenário (instância dos casos de uso)

1. User A presses the Up floor button at floor 3 to request an elevator. User A wishes to go to floor 7.
2. The Up floor button is turned on.
3. An elevator arrives at floor 3. It contains User B, who has entered the elevator at floor 1 and pressed the elevator button for floor 9.
4. The elevator doors open.
5. The timer starts.
   User A enters the elevator.
6. User A presses the elevator button for floor 7.
7. The elevator button for floor 7 is turned on.
8. The elevator doors close after a timeout.
9. The Up floor button is turned off.
10. The elevator travels to floor 7.
11. The elevator button for floor 7 is turned off.
12. The elevator doors open to allow User A to exit from the elevator.
13. The timer starts.
    User A exits from the elevator.
14. The elevator doors close after a timeout.
15. The elevator proceeds to floor 9 with User B.

Modelagem

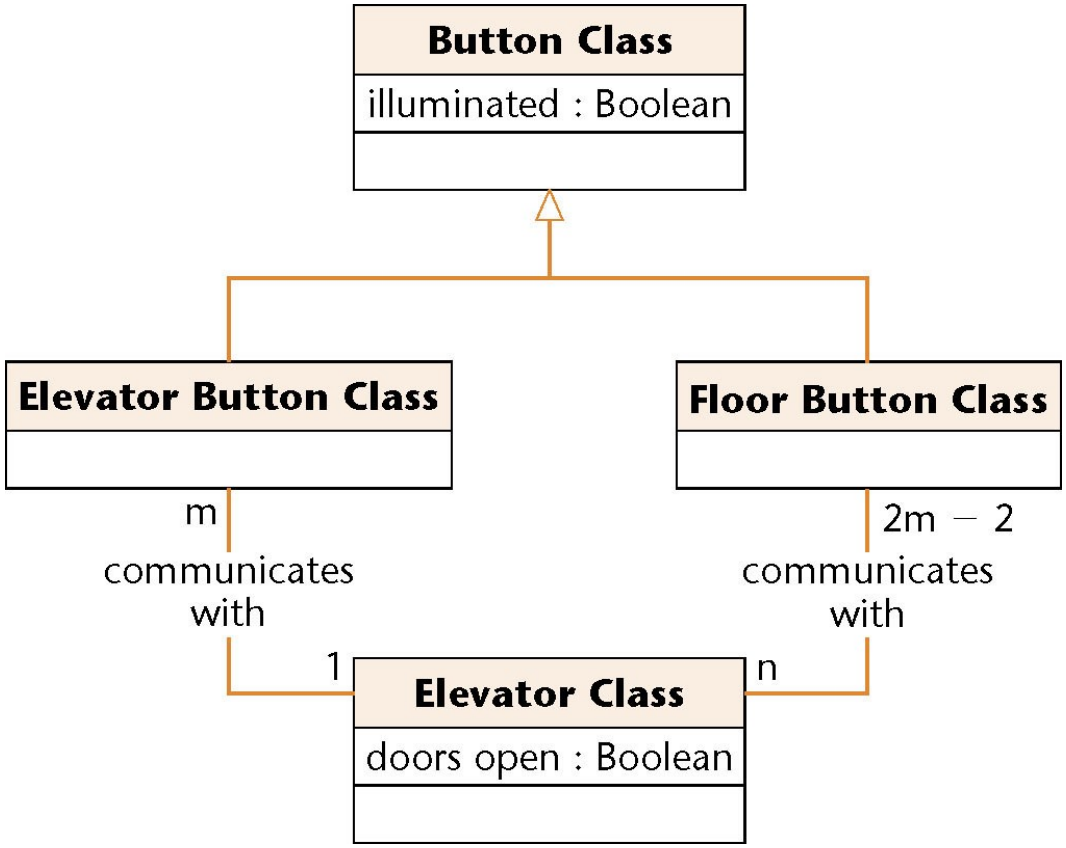
- Identificação de substantivos
- Buttons in elevators and on the floors control the movement of n elevators in a building with m floors. Buttons illuminate when pressed to request the elevator to stop at a specific floor; the illumination is canceled when the request has been satisfied. When an elevator has no requests, it remains at its current floor with its doors closed

Modelagem

- Identificação de substantivos
- Buttons in elevators and on the floors control the movement of n elevators in a building with m floors. Buttons illuminate when pressed to request the elevator to stop at a specific floor; the illumination is canceled when the request has been satisfied. When an elevator has no requests, it remains at its current floor with its doors closed

- Classes candidatas: Elevador e Botão
- Subclasses candidatas: BotãoElevador e BotãoAndar

Modelagem



Problema: Quem controla o conjunto de elevadores?

Uso de uma classe de controle

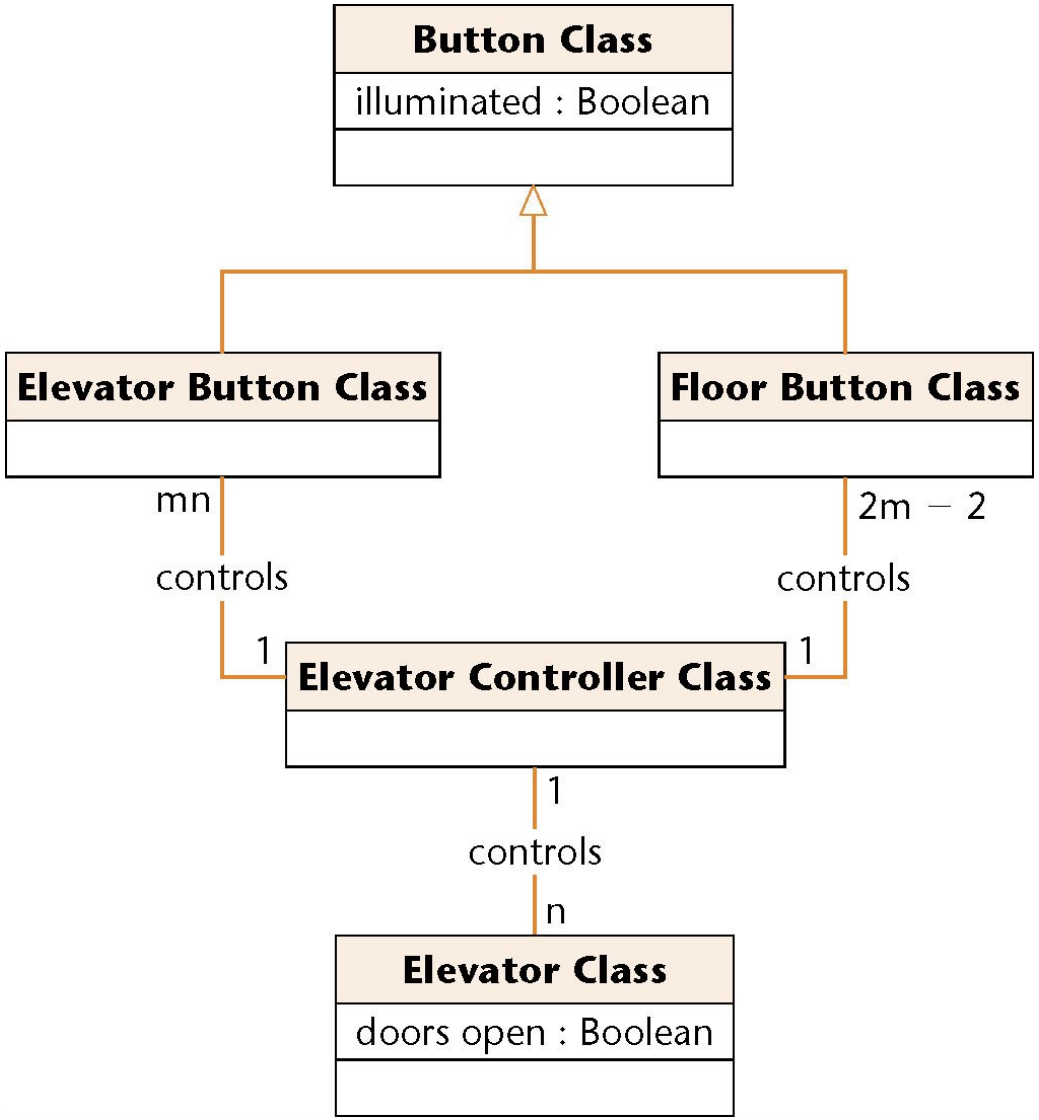


Diagrama de estados

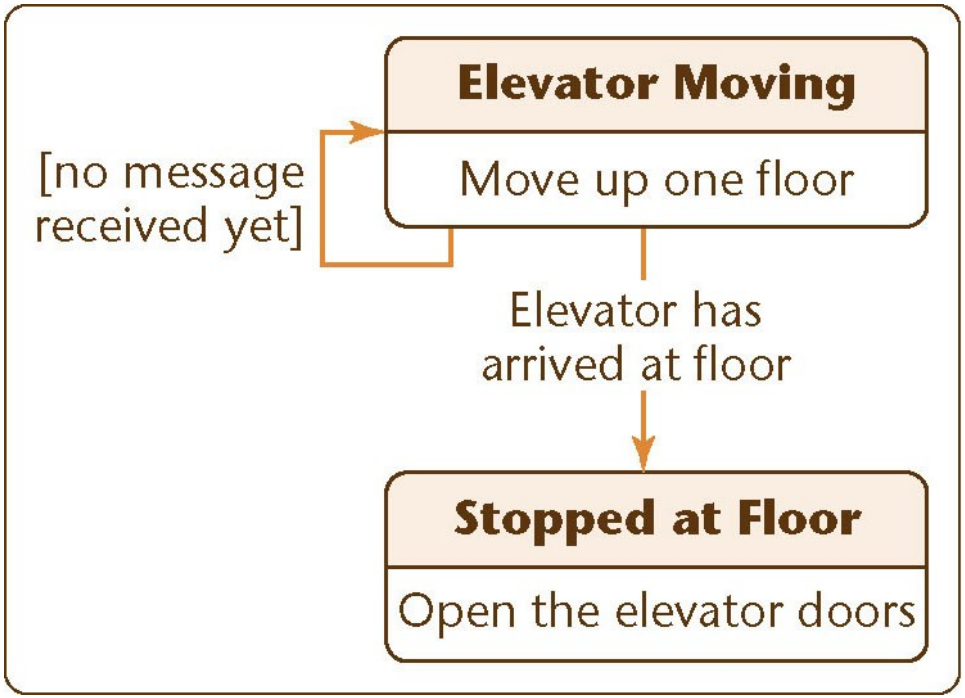


Diagrama de estados mais elaborado