Projeto Integrador 3
Desenvolvimento de Sistemas Orientado a Objetos

Especificação dos Requisitos de Software (ERS)

09/09/2016

Prof. Fernando Tsuda

Objetivos

- Apresentar os itens necessários que deverão constar no documento de especificação dos requisitos de software

Especificação dos Requisitos de Software

- Documento criado quando há a necessidade de uma descrição de todos os aspectos do software a ser construído.
- Elaborado antes do início da construção do software.
- Artefatos UML (casos de uso, diagrama de classes, protótipos de telas) e MER compõem o documento como anexos e auxiliam o entendimento do sistema que será desenvolvido.

- Normalmente, o documento é resultado da fase de análise dos requisitos, e direciona as atividades dos programadores responsáveis pela construção.

Itens do documento

Para esta disciplina, focar o preenchimento dos itens que estão marcados em negrito nas próximas seções.

Estrutura do ERS (1/5)

- Sumário
- Histórico de revisões

1. Introdução
    1. **Propósito**
    2. Convenções do documento
    3. Público-alvo e sugestões de leitura
    4. **Escopo do projeto**
    5. Referências

Estrutura do ERS (2/5)

2. Descrição geral
    1. **Perspectiva do produto**
    2. **Característica do produto**
    3. **Classes de usuários e características**
    4. **Ambiente operacional**
    5. **Restrições de projeto e implementação**
    6. Documentação para usuários
    7. **Hipóteses (ou premissas) e dependências**

Estrutura do ERS (3/5)

3. Característica do sistema
    1. **Característica do sistema 1**
    2. **Característica do sistema 2**
    3. **...**
    4. **Característica do sistema N**

Estrutura do ERS (4/5)

4. Requisitos de interfaces externas
   1. **Interfaces do usuário**
   2. **Interfaces de hardware**
   3. **Interfaces de software**
   4. **Interfaces de comunicação**
5. Outros requisitos não funcionais
   1. Necessidades de desempenho
   2. **Necessidades de segurança (security – Proteção contra ataques intencionais)**
   3. Necessidades de proteção (safety – Proteção contra acidentes)
   4. **Necessidades de disponibilidade de software para o usuário**
   5. **Atributos de qualidade de software**
      - ***Definir quais são os critérios para que a entrega do software seja aceita (ex: % de testes executados com sucesso)***
      - ***Definir uma lista de como testar as funcionalidades.***

Estrutura do ERS (5/5)

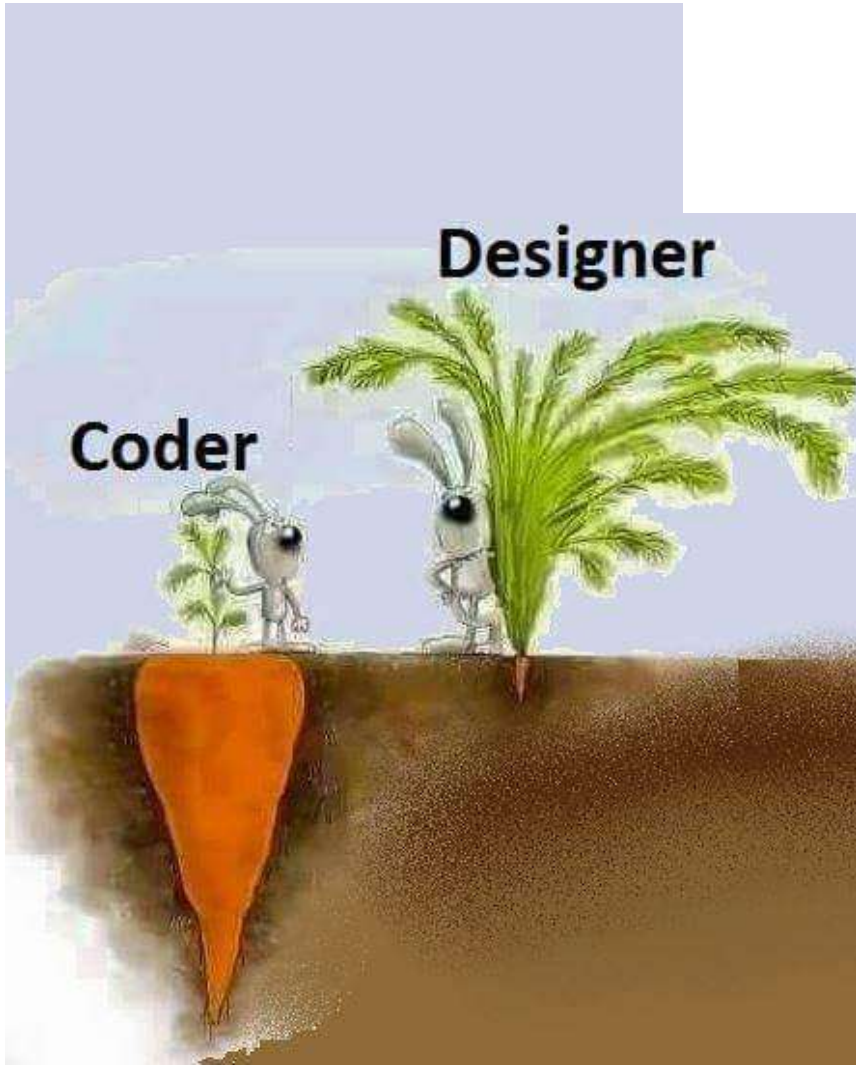
6. Outros requisitos
   - Apêndice A: Glossário
   - **Apêndice B: Modelos de análise**
     - ***Wireframes das telas, Descrição e diagrama dos casos de uso, Diagrama de classes, etc.***
   - Apêndice C: Lista de problemas

Projeto de Interfaces das Telas

Design de interfaces

- Tem como objetivo criar um meio pelo qual o usuário possa operar o sistema;
- Normalmente, é a parte do sistema que transmite a “percepção de qualidade” ao usuário.
- Regras de ouro de Theo Mandel:
  - Deixar o usuário no comando;
  - Reduzir a carga de memória do usuário;
  - Tornar a interface consistente.

Design de interfaces



Etapas para definir a interface

- Análise do usuário
  - Entrevistas;
  - Informações sobre os perfis dos usuários (nível técnico, nível de educação formal, meio preferencial de treinamento, faixa etária, gênero, etc).
- Análise das tarefas
  - Qual trabalho será realizado pelos usuários?
  - Qual o fluxo do trabalho?
  - Quais as consequências de um erro na realização das tarefas?

Usabilidade

- Em sistemas computacionais, define a facilidade que o usuário tem para usar o sistema.
- ISO 9241 – Ergonomia de software de escritório

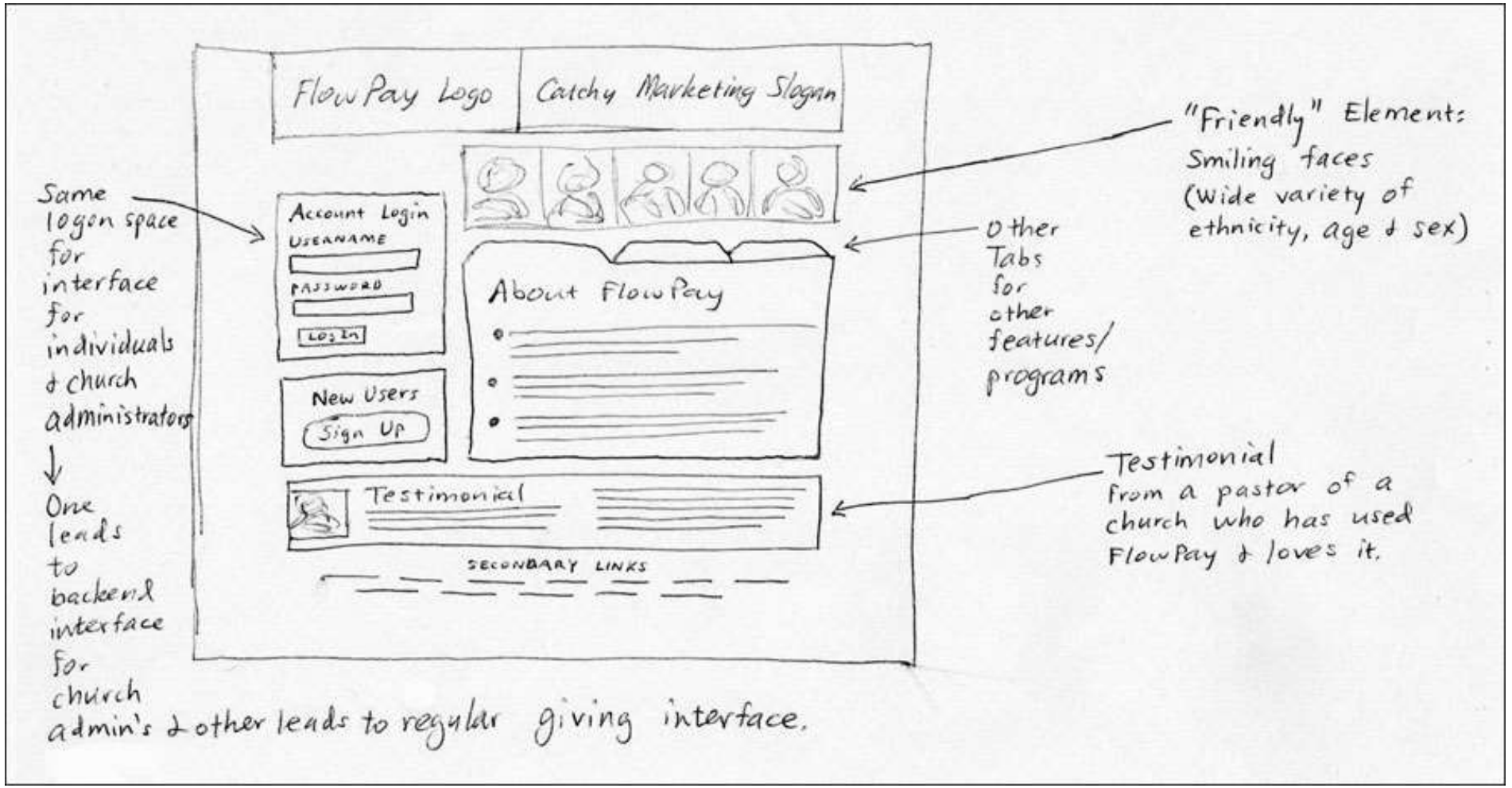
  “Medida pela qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos com efetividade, eficiência e satisfação em um contexto de uso específico.”
- UI: User Interface
- UX: User eXperience

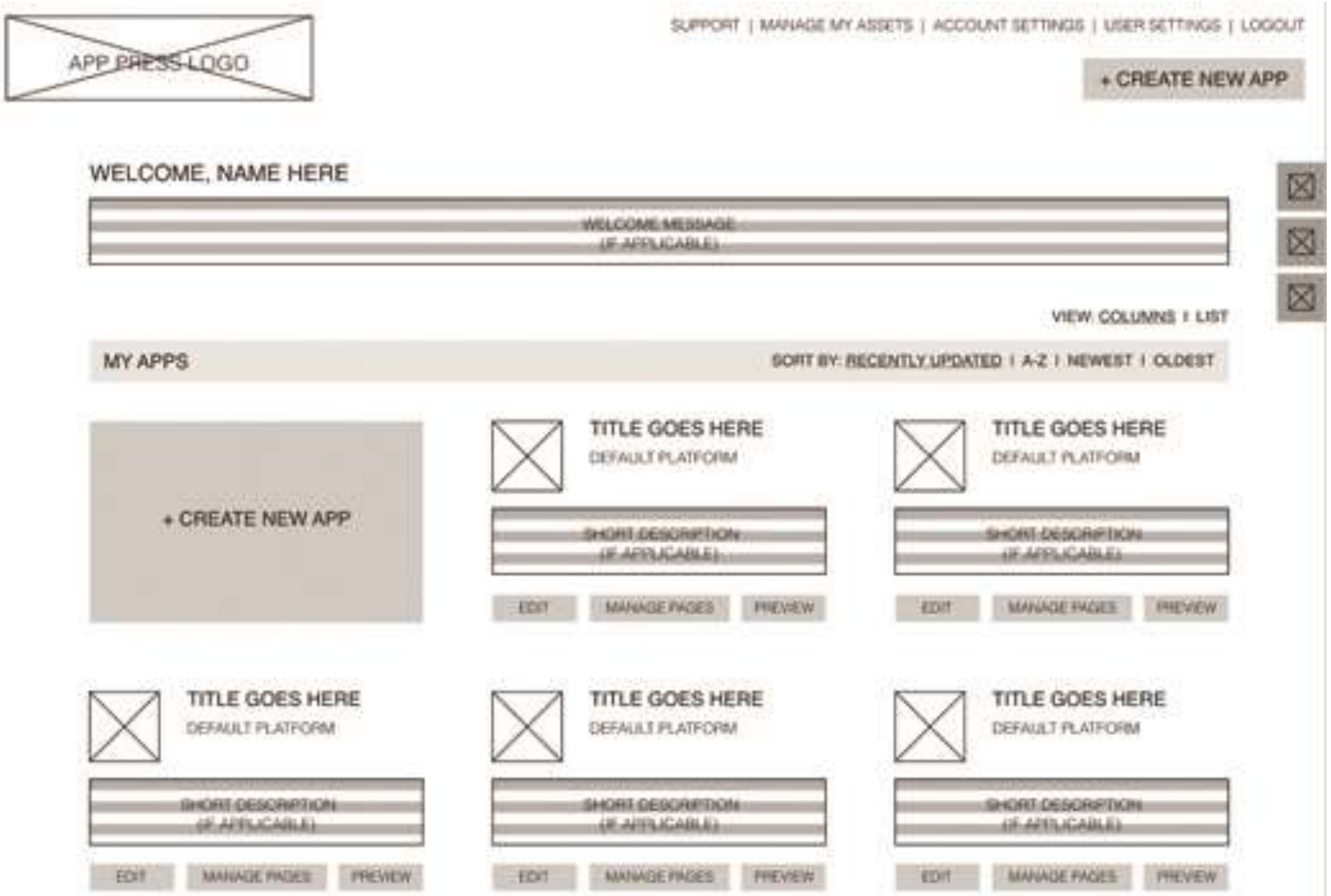
Usabilidade

- Exemplos de elementos que afetam usabilidade:
  - Tamanho das letras;
  - Contraste de cores;
  - Tamanho de botões;
  - Uso de dicas (tooltips e placeholders);
  - Comportamentos alternativos (ex: textos alternativos às imagens que não são apresentadas);
  - Textos alternativos para “leitores de tela” (usado por deficientes visuais).

Wireframes de telas

- Esboço inicial da forma como organizar os elementos gráficos na tela, de forma a facilitar a apresentação da ideia para outras pessoas;
- Em sistemas computacionais, normalmente envolve o desenho das telas de interface e a sequência de navegação entre as telas, sem a lógica de negócio;
- O wireframe também pode ser referenciado como **mockup**.


FlowPay Logo
Catchy Marketing Slogan
Account Login
USERNAME
PASSWORD
LOG IN
New Users
Sign Up
About FlowPay
Testimonial
SECONDARY LINKS
Same logon space for interface for individuals & church administrators
One leads to backend interface for church admin's & other leads to regular giving interface.
Other Tabs for other features/ programs
"Friendly" Elements: Smiling faces (Wide variety of ethnicity, age & sex)
Testimonial from a pastor of a church who has used FlowPay & loves it.


APP PRESS LOGO
SUPPORT | MANAGE MY ASSETS | ACCOUNT SETTINGS | USER SETTINGS | LOGOUT
+ CREATE NEW APP
WELCOME, NAME HERE
WELCOME MESSAGE (IF APPLICABLE)
VIEW: COLUMNS | LIST
MY APPS
SORT BY: RECENTLY UPDATED | A-Z | NEWEST | OLDEST
+ CREATE NEW APP
TITLE GOES HERE
DEFAULT PLATFORM
SHORT DESCRIPTION (IF APPLICABLE)
EDIT
MANAGE PAGES
PREVIEW


Some Sort of H1 Tag and a Cool Tag Line
My Wireframe above becomes this...
IDENTITY
MAINTAIN USER SECURITY

Ferramentas

- Lápis e Papel
- Photoshop
- Microsoft Visio
- Moqups (moqups.com)
- Cacoo (cacoo.com)
- Marvel App (marvelapp.com) → Permite criar protótipos de apps móveis
- Outros
  - http://www.creativebloq.com/wireframes/top-wireframing-tools-11121302
  - http://mashable.com/2010/07/15/wireframing-tools/

Referências usabilidade

- http://www.uxdesign.blog.br/usabilidade/10-licoes-de-usabilidade-de-steve-krug/
- https://www.nickkolenda.com/user-experience/

Casos de Uso

Casos de uso

- Ferramenta de análise usada para descrever as diversas formas de interações entre os usuários (humanos ou máquinas) – ou atores – com o sistema que será desenvolvido;
  - Pela definição de Ivan Jacobson, é uma ferramenta para definir o comportamento entre o que existe fora do sistema e o que deve ser realizado pelo sistema.
- Através das informações contendo os cenários de utilização do sistema, é possível identificar os passos necessários para atingir um objetivo específico.

Casos de uso

- Normalmente, a descrição de um caso de uso pode ser feita seguindo-se o modelo abaixo:
  - Nome do caso de uso
  - Descrição breve
  - Ator primário
  - Atores secundários
  - Pré-condições
  - Descrição do fluxo principal
  - Descrição dos fluxos ou etapas alternativas
  - Pós-condições
  - Regras de negócio associadas
  - Lista dos requisitos funcionais e não funcionais associados

http://www.galeote.com.br/blog/2010/08/template-para-descrio-de-caso-de-uso/

Casos de uso - Exemplo (1/2)

- **001 - Cadastrar cliente**
- Descrição: Funcionalidade onde um novo cliente realiza seu cadastro no sistema.
- Ator primário: Cliente da loja.
- Atores secundários: não há.
- Pré-condições:
  1. Cliente não cadastrado no sistema
- Fluxo principal:
  1. Cliente acessa a tela inicial e clica no botão cadastrar
  2. Sistema apresenta tela contendo formulário de cadastro
  3. Cliente preenche os dados e clica no botão salvar (EX01, ALT01)
  4. Sistema salva os dados e apresenta mensagem de sucesso

Casos de uso - Exemplo (2/2)

- Fluxos alternativos
  - EX01 – Cliente não preenche todos os campos obrigatórios
    1. Sistema reapresenta quais campos devem ser preenchidos.
  - ALT01 – Cliente desiste do cadastro
    1. Cliente deve clicar no botão cancelar
    2. Sistema apresenta tela inicial
- Pós-condições:
  1. Novo cliente cadastrado no sistema
- Regras de negócio associadas
  - Cliente maior de 18 anos; nome, CPF e data de nascimento obrigatórios; E-mail válido obrigatório.
- Requisitos funcionais e não funcionais
  - Gravar dados do cliente no banco de dados; dados gravados devem ser protegidos; dados gravados devem ser consistentes e duráveis.

Casos de uso - Exemplo

A) Numerar os casos

B) Nomear usando verbo no infinitivo + substantivo

- **001 - Cadastrar cliente**
- Descrição: Funcionalidade onde um novo cliente realiza seu cadastro no sistema.
- Ator primário: Cliente da loja.
- Atores secundários: não há.
- Pré-condições:
  1. Cliente não cadastrado no sistema
- Fluxo principal:
  1. Cliente acessa a tela inicial e clica no botão cadastrar
  2. Sistema apresenta tela contendo formulário de cadastro
  3. Cliente preenche os dados e clica no botão salvar (EX01, ALT01)
  4. Sistema salva os dados e apresenta mensagem de sucesso

C) Se a informação não se aplicar, preencher com “não há” ou “não se aplica”

D) Numerar a sequência do fluxo.

Diagrama de casos de uso

- Diagrama da UML (Unified Modeling Language) de alto nível que permite representar visualmente a interação entre os atores e o sistema;
- Facilita a verificação da quantidade de funcionalidades que o sistema possui.
- Notar que somente através do diagrama não é possível identificar os detalhes do caso de uso. Esse detalhamento deve ser feito na descrição textual.

Diagrama de casos de uso

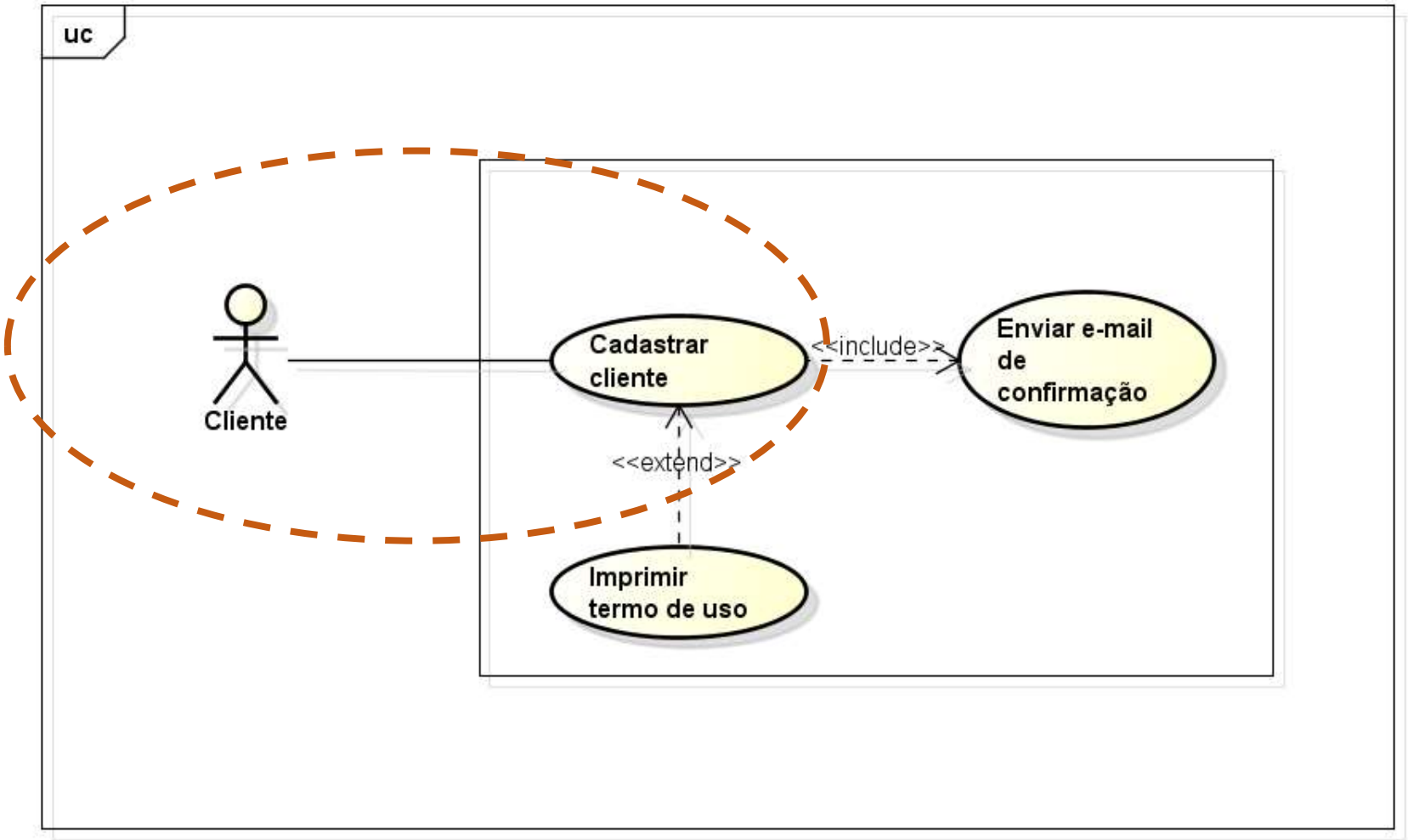


Diagrama de Classes

Diagrama de classes

- Diagrama que oferece uma visão estrutural das entidades essenciais que compõem um sistema;
- É um diagrama estático, ou seja, é usado para mostrar as características das entidades e como elas se relacionam;
- Não serve para mostrar informações de natureza dinâmica.

Classes e Objetos

- Classe: estrutura da Programação Orientada a Objetos (POO) usada para modelar as características e comportamentos de uma entidade real.
  - Características do objeto são representadas pelos atributos;
  - Comportamentos são representados pelos métodos (ou operações).
- Objeto: entidade criada a partir de uma instância da classe.

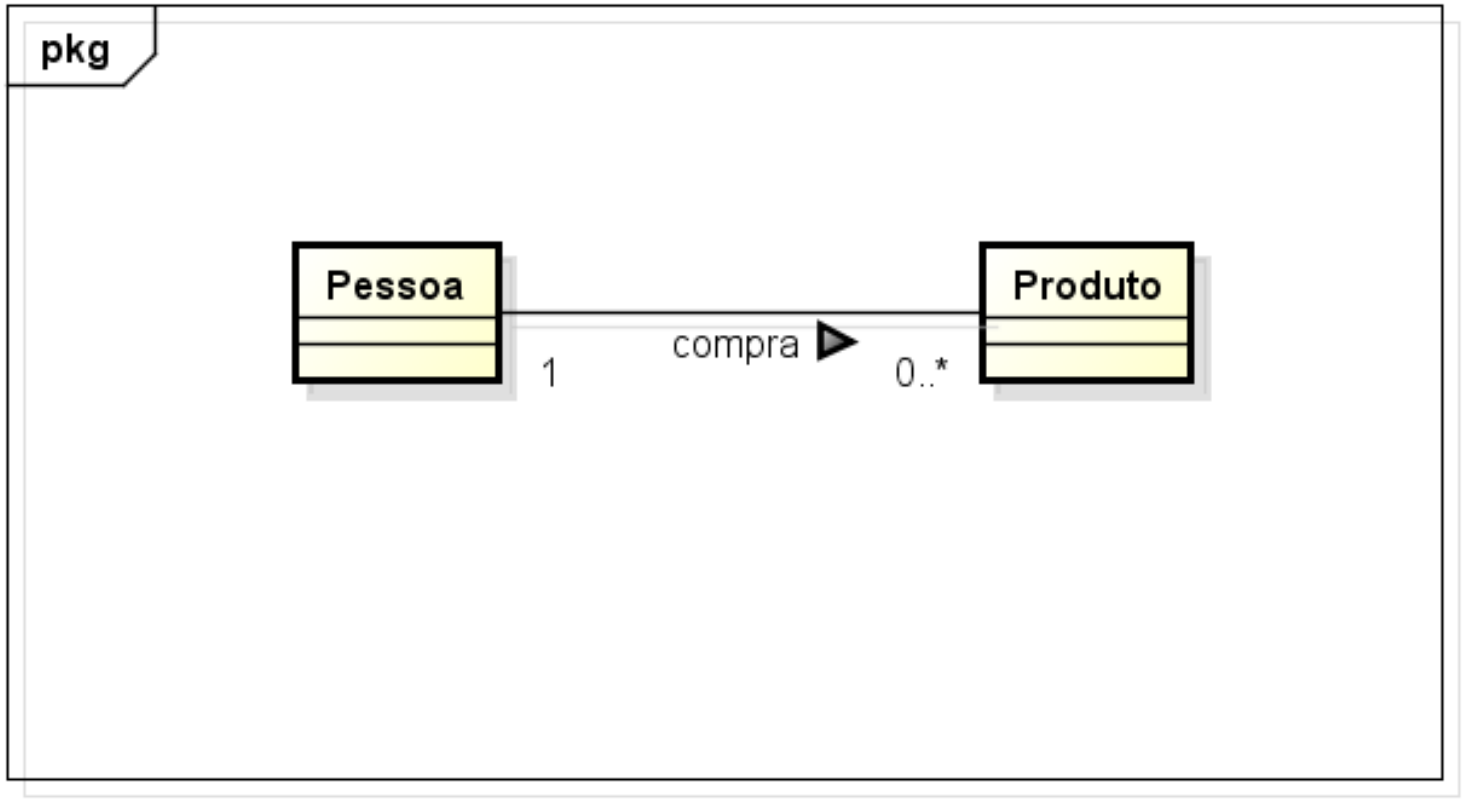
Classes e Objetos – Exemplo classe Java

```
public class Pessoa {
        /* Atributos */
        private String nome; private Date dataNascimento;
        /* Construtora */
        public Pessoa() { }
        /* Getter e Setter */
        public String getNome() { return nome; }
        public void setNome(String nome) { this.nome = nome; }
        public Date getDataNascimento() { return dataNascimento; }
        public void setDataNascimento(Date data) { dataNascimento = data; }
        /* Métodos (ou operações) */
        public String falar() { ... }
        public void mover() { ... }
        public void ouvir(String som) { ... }
}
```

Conceitos importantes em POO

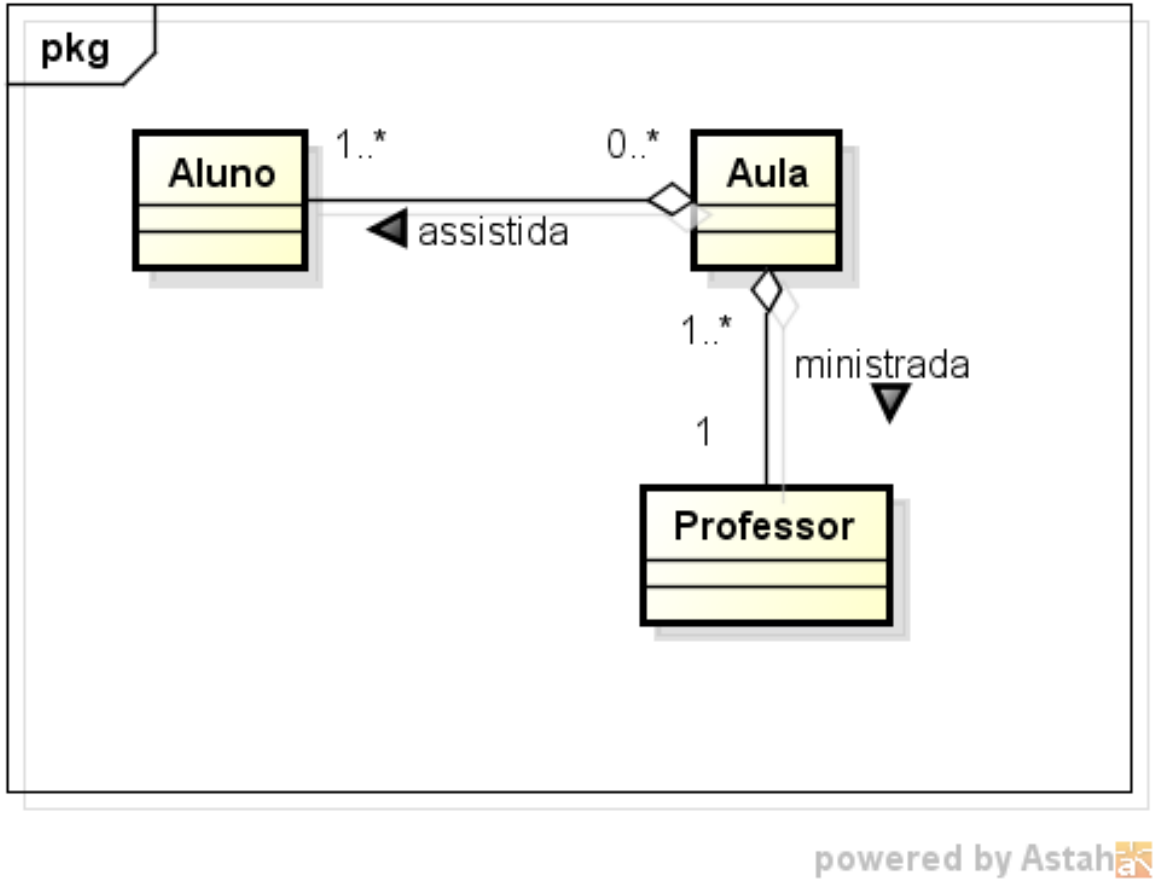
- Herança: conceito onde uma classe pode “herdar” algumas características de outra classe e especializar outras;
- Polimorfismo: capacidade dos objetos e seus métodos serem referenciados de várias maneiras;
- Encapsulamento: Separação de aspectos internos e externos de um objeto.

Relações entre as classes



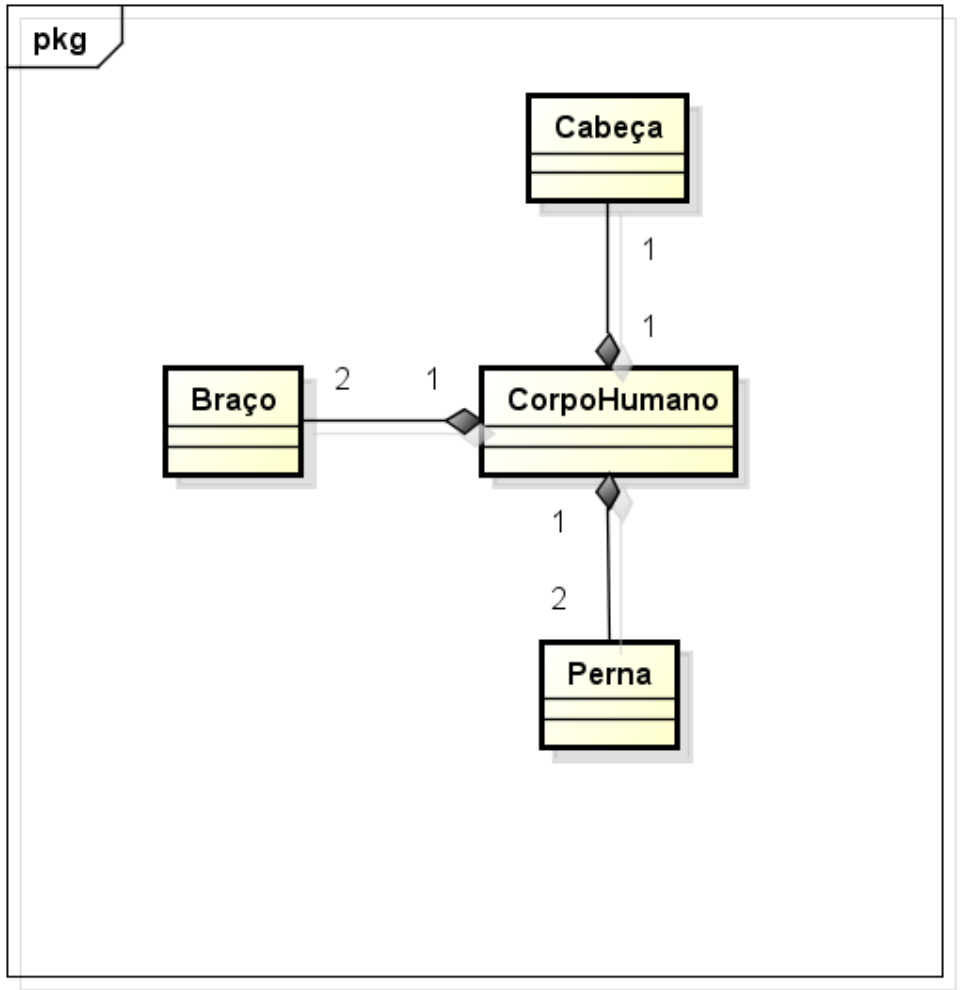
Relações entre as classes

- Agregação
  - Relação do tipo “Todo” possui “Parte”.
  - O “Todo” e a “Parte” existem separadamente



Relações entre as classes

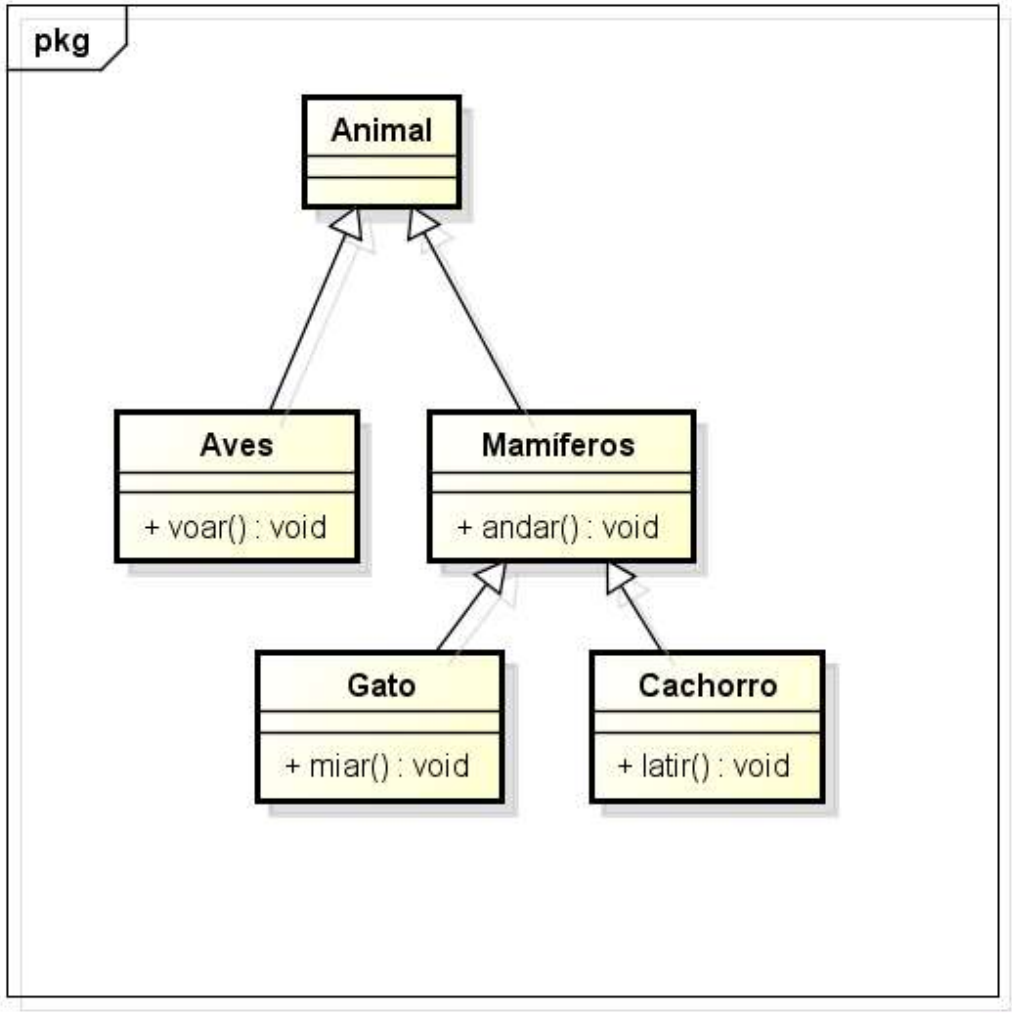
- Composição
  - Relação do tipo “Todo” possui “Parte”.
  - A “Parte” não existe sem o “Todo”



http://imasters.com.br/artigo/18901/uml/uml-composicao-x-agregacao/

Relações entre as classes

- Herança
  - Representa uma relação do tipo “is-a” (“é um”) entre as classes.



Referências

- https://docs.oracle.com/javase/tutorial/java/concepts/index.html
- http://www.devmedia.com.br/modelo-entidade-relacionamento-mer-e-diagrama-entidade-relacionamento-der/14332