PROJETO DE SOFTWARE - CONSTRUÇÃO

DESAFIOS DO DESIGN

- Projeto de Software é um problema perverso
- O projeto de software é um processo desordenado (mesmo que produza um resultado ordenado)
- O projeto de software envolve equilíbrio e prioridades
- O projeto de software envolve restrições
- O projeto de software não é determinístico
- O projeto de software é um processo heurístico
- O projeto de software possui comportamento emergente

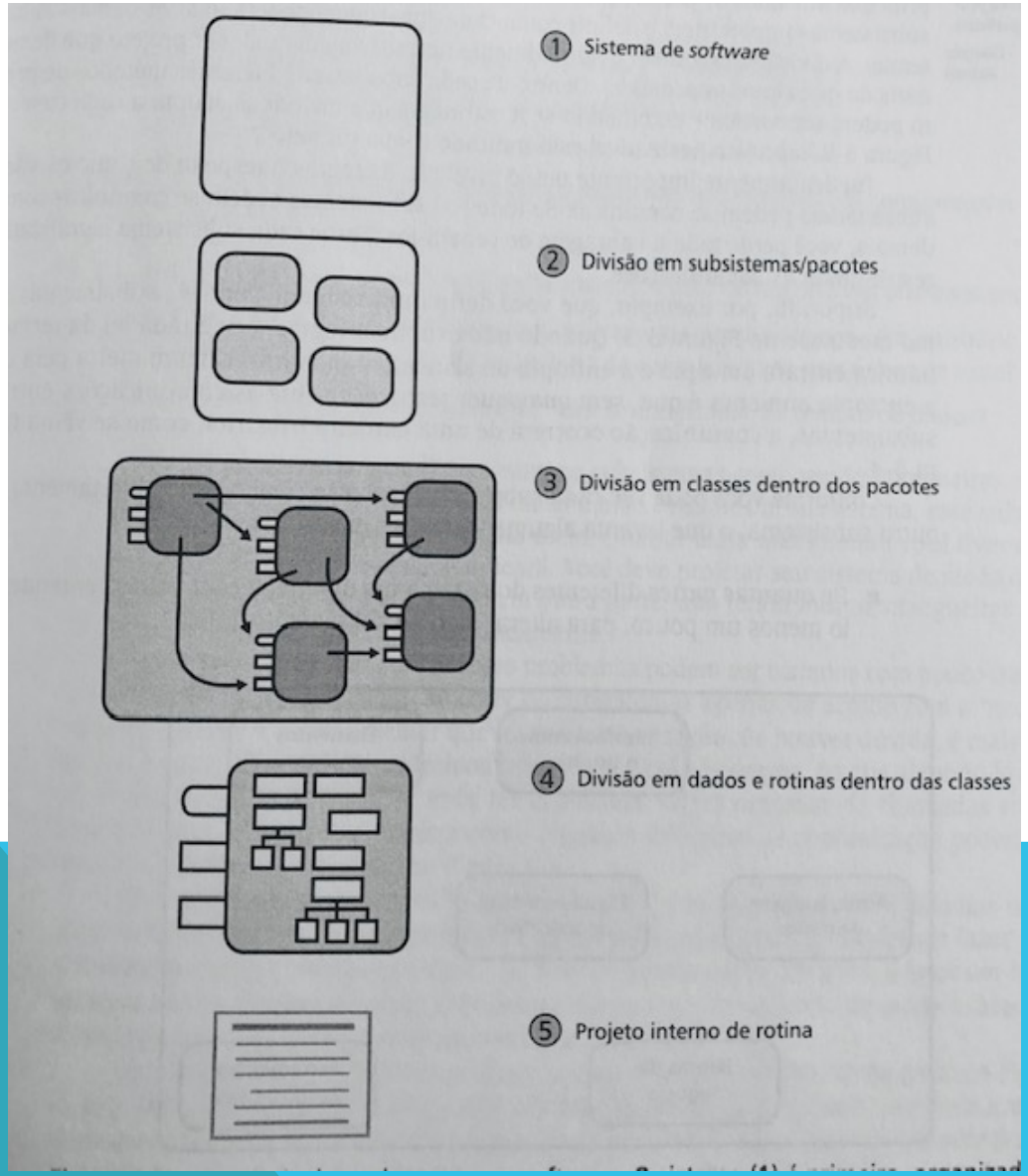
PRINCIPAIS CONCEITOS

- Controle da Complexidade
- Problemas acidentais
- Problemas essenciais

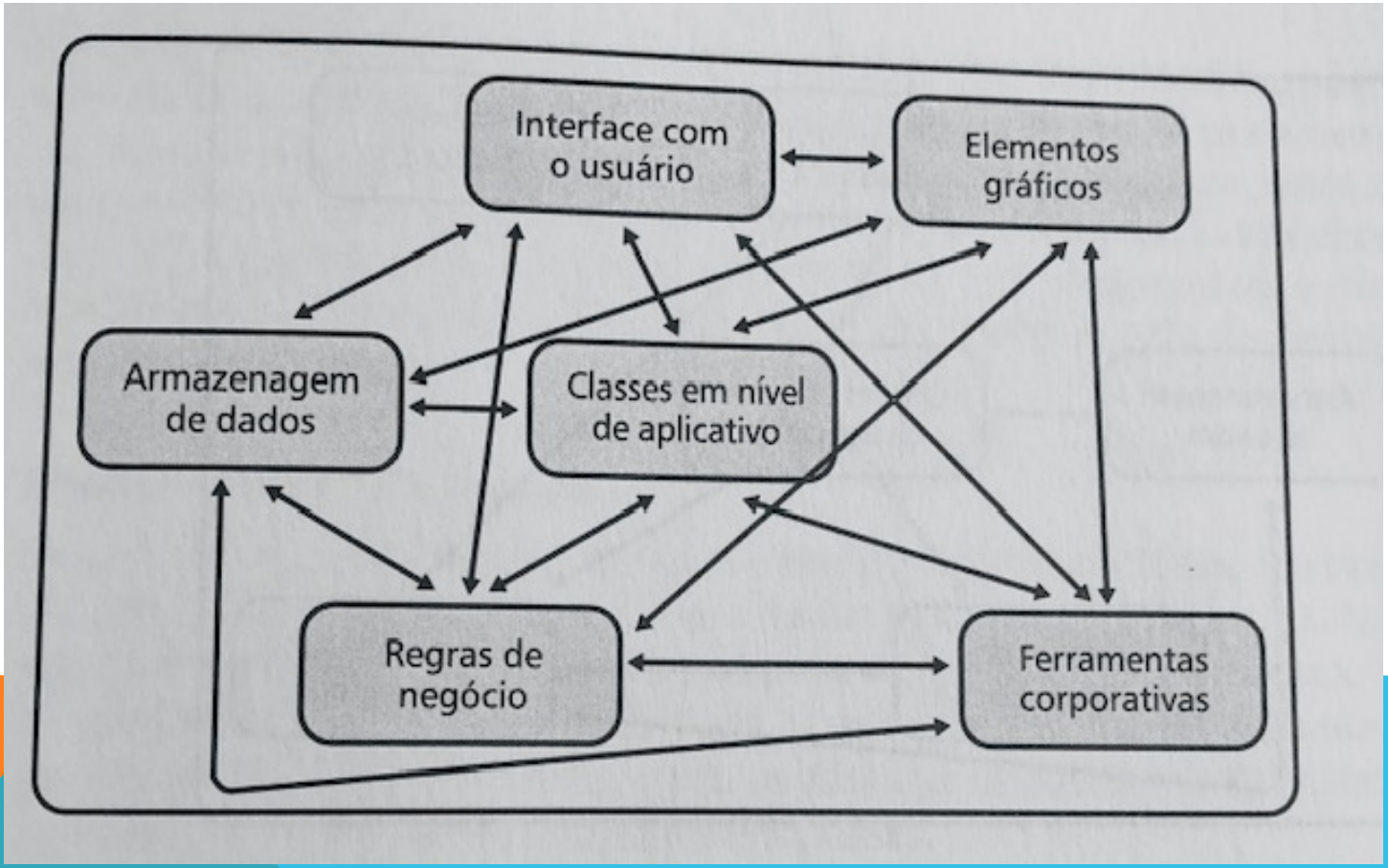
CARACTERÍSTICAS DESEJÁVEIS

- Complexidade mínima
- Facilidade de manutenção
- Baixo acoplamento
- Extensibilidade
- Reutilização
- Objetividade
- Estratificação
- Técnicas-padrão

NÍVEIS DE PROJETO



DIVISÃO EM SUBSISTEMAS

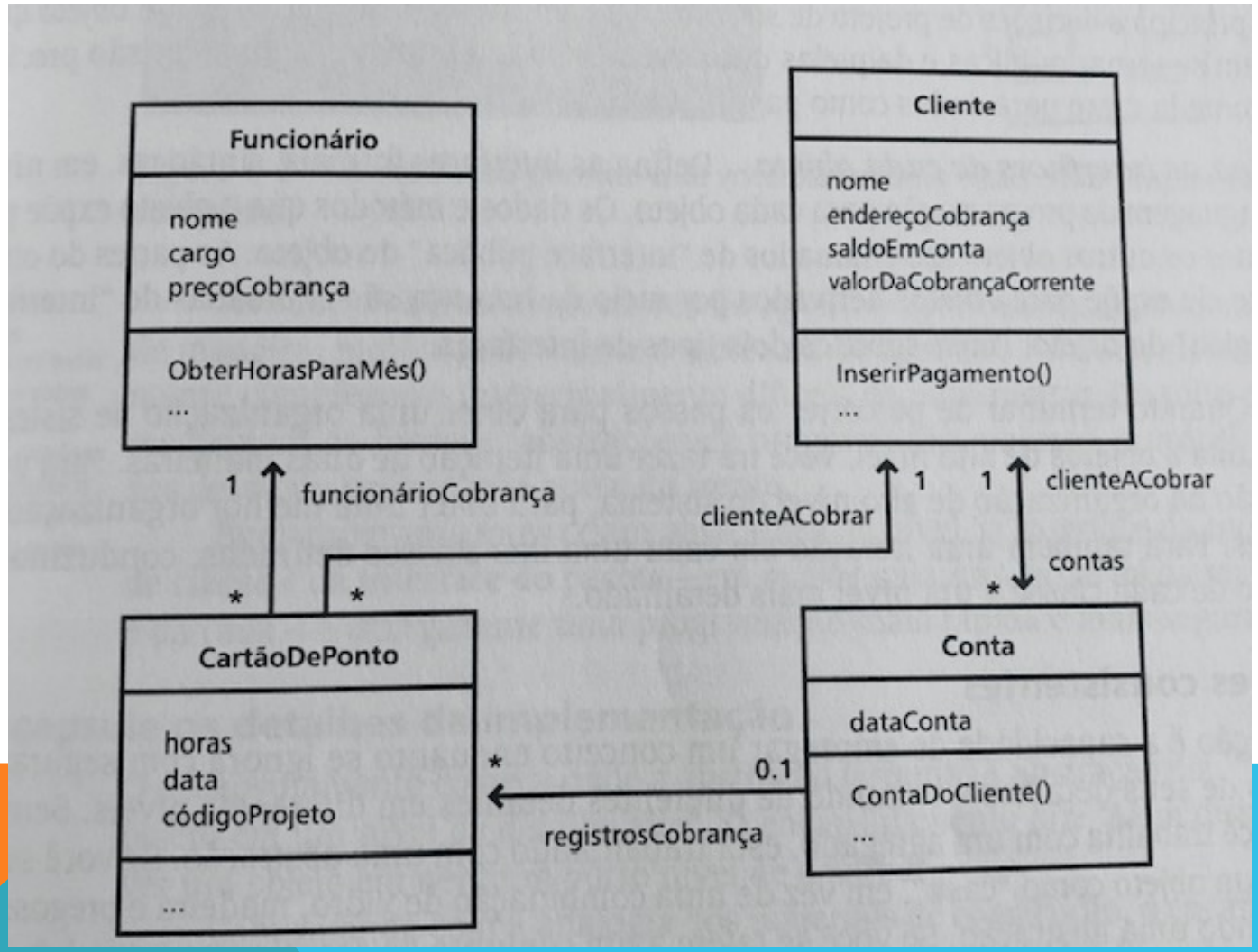


HEURÍSTICAS

ENCONTRE OBJETOS DO MUNDO REAL

- Identifique os objetos e seus atributos (métodos e dados).
- Determine o que pode ser feito em cada objeto.
- Determine o que cada objeto pode fazer com outros objetos.
- Determine as partes (públicas e privadas) de cada objeto que serão visíveis para outros objetos
- Defina a interface pública de cada objeto.

IDENTIFIQUE OS OBJETOS E SEUS ATRIBUTOS



CRIE ABSTRAÇÕES CONSISTENTES

Abstração é a capacidade de empregar um conceito enquanto se ignora com segurança alguns de seus detalhes - tratando de diferentes detalhes em diferentes níveis. Sempre que você trabalha com um agregado, está trabalhando com uma abstração. Se você se refere a um objeto como “casa”, em vez de uma combinação de vidro, madeira e pregos, está fazendo uma abstração. Se você se refere a um conjunto de casas como “cidade”, esta fazendo outra abstração.

ENCAPSULE OS DETALHES DA IMPLEMENTAÇÃO

O encapsulamento começa onde a abstração termina. A abstração diz: “Você pode ver um objeto em um nível de detalhe alto”. O encapsulamento diz: “Além disso, você não pode ver um objeto em nenhum outro nível de detalhe”.

HERDE - QUANDO A HERANÇA SIMPLIFICA O PROJETO DE SOFTWARE

A vantagem da herança é que ela funciona em sinergia com a noção de abstração. A abstração trata com objetos em diferentes níveis de detalhe.

A herança simplifica a programação porque você escreve uma rotina genérica para tratar de tudo que diz respeito às propriedades genéricas e depois escreve rotinas específicas para tratar de operações específicas nos tipos específicos de objetos.

Algumas operações, como Abrir() ou Entrar(), podem se aplicar independentemente de a porta ser sólida, interior, exterior, blindada, dupla ou de vidro corrediça. A capacidade de uma linguagem de suportar operações como Abrir() ou Fechar() sem saber, até o momento da execução, com que tipo de porta você está tratando, é chamada de “polimorfismo”.

OCULTAMENTO DE INFORMAÇÕES

O ocultamento de informações faz parte da base do projeto estruturado e do projeto orientado a objetos. No projeto estruturado, a noção de "caixas pretas" é proveniente do ocultamento de informações. No projeto orientado a objetos, ele dá origem aos conceitos de encapsulamento e modularidade, e é associado ao conceito de abstração. O ocultamento de informações é uma das ideias embrionárias no desenvolvimento de software. É uma heurística particularmente poderosa pois, começando com seu nome e em todos os seus detalhes, ele acentua o ocultamento da complexidade

IDENTIFIQUE AS ÁREAS DE PROVÁVEL ALTERAÇÃO

Acomodar as alterações é um dos aspectos mais desafiadores do bom projeto de programas. O objetivo é isolar áreas instáveis para que o efeito de uma alteração fique limitado a uma rotina, classe ou pacote.

1. Identifique os itens que provavelmente irão mudar.
2. Separe os itens que provavelmente irão mudar.
3. Isole os itens que provavelmente irão mudar.

MANTENHA O ACOPLAMENTO BAIXO

O acoplamento descreve o quanto uma classe ou rotina está relacionada com outras classes ou rotinas. O objetivo é criar classes e rotinas com relações pequenas, diretas, visíveis e flexíveis com outras classes e rotinas, o que é conhecido como “baixo acoplamento”.

Um bom acoplamento entre módulos é baixo o suficiente para que um módulo possa ser facilmente usado por outros módulos.

PROCURE PADRÕES DE PROJETO COMUNS

Adapter

Bridge

Decorator

Façade

Factory Method

Observer

Singleton

Strategy

Template Method

OUTRAS HEURÍSTICAS

- Tenha como objetivo uma coesão forte
- Construa hierarquias
- Formalize os contratos de classe
- Atribua responsabilidades
- Desenvolva o projeto pensando nos testes
- Evite falhas
- Escolha conscientemente o momento da vinculação
- Crie pontos centrais de controle
- Considere o uso da força bruta
- Desenhe um diagrama
- Mantenha seu projeto modular

PONTOS-CHAVE

- O principal imperativo técnico do software é o controle da complexidade. Isso é bastante facilitado por um foco do projeto na simplicidade.
- A simplicidade é obtida de duas maneiras gerais: minimizando a quantidade de complexidade essencial com que o cérebro de alguém precisa lidar em dado momento e impedindo que a complexidade acidental prolifere desnecessariamente.
- Projeto de software é heurística. O apego dogmático a uma única metodologia prejudica a criatividade e seus programas.
- O bom projeto de software é iterativo; quanto mais possibilidades de projeto você tentar, melhor será seu projeto final.
- O ocultamento de informações é um conceito particularmente valioso. Perguntar “o que eu devo ocultar?” resolve muitos problemas difíceis de projeto de software.