Projeto Detalhado de Software

GRASP
Criador, Especialista na Informação,
Controlador, Acoplamento e Coesão

Camilo Almendra | camilo@es.ufc.br | https://sites.google.com/a/es.ufc.br/camilo/

Informações

REFERÊNCIAS

- **Utilizando UML e Padrões**. LARMAN, Craig. 3a Edição. Capítulo 17 e 18.
  - *Atenção para o capítulo 18, que faz um passo a passo de iniciação de um projeto.*
- **Princípios, padrões e práticas ágeis em C#**. MARTIN, Robert. MARTIN, Micah. 1ª edição. Capítulo 14.
- **Code Complete**. MCCONNELL, Steve. 2a Edição. Capítulos 5 e 7.
- Lei de Demétrio (“Tell, Don't Ask”)

  http://msdn.microsoft.com/pt-br/magazine/cc947917.aspx

LABORATÓRIOS

- Laboratório 2.1 – Modelagem de cenários
- Laboratório 2.2 – Desacoplando UI e Lógica
-

Introdução

- Após os primeiros anos da programação orientada a objetos, tornou-se popular raciocinar em termos de **responsabilidades** no projeto de sistemas OO

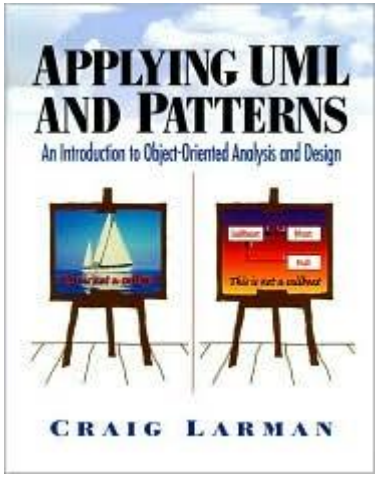
- **Responsabilidades, Papéis e Colaborações** formam a tríade do chamado Projeto Guiado por Responsabilidades (PGR)

Projeto Guiado por Responsabilidades

- Metáfora do PGR
  - Objetos possuem responsabilidades
  - Objetos colaboram entre si
  - Similar ao relacionamento entre pessoas
- Tipos de responsabilidades de um objeto
  - **FAZER ALGO (comportamento)**
  - **SABER ALGO (informações encapsuladas)**

GRASP

- GRASP (General Responsibility Assignment Software Patterns)
  - Padrões básicos para atribuir responsabilidades
  - Abordagem metódica para projeto OO
  - Princípios amplamente usados
- 09 Padrões



Criador	**Especialista na Informação**	**Acoplamento**
Controlador	**Coesão**	**Polimorfismo**
Indireção	**Invenção Pura**	**Variações Protegidas**

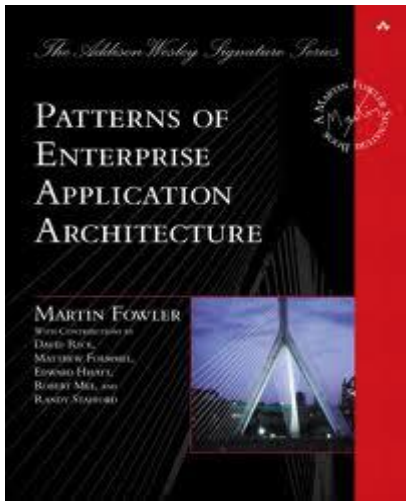
Padrão

- Padrão descreve uma solução para um problema, em um determinado contexto
- Padrões são uma forma de compartilhar soluções populares para problemas recorrentes
- Formato básico do Padrão
  - Nome
  - Problema / Contexto
  - Solução

Padrões

- Projetar com ajuda de Padrões é:
  - Reusar soluções comprovadas, através da identificação de problemas e contextos semelhantes ou equivalentes;
  - Tomar decisões de projeto seguindo diretrizes/princípios;
  - Se inspirar, mas não se amarrar.
- Exemplos de catálogos →

GoF



PoEAA

GRASP foca em didática

- Entender os padrões GRASP é o passo básico no aprendizado
- Existem outros catálogos de padrões (GoF, PoEAA) mais completos e mais complexos
- Exemplo:
  - *Strategy*: padrão que desacopla Informação e Algoritmo
  - *Factory*: padrão que desacopla a Criação de objetos do domínio

GRASP

- **Diretrizes e Padrões**
  - Baixo hiato representacional (diretriz)
  - Criador
    - Foco no início do ciclo de vida dos objetos
  - Especialista na Informação
    - Foco no encapsulamento
  - Separação da Visão e Modelo (diretriz)
  - Controlador
    - Foco em separar interação com usuário e lógica de negócios
  - Acoplamento
    - Foco no impacto da mudança
  - Coesão
    - Foco em diminuir a complexidade

Baixo Hiato Representacional

DIRETRIZ

- Hiato = Lacuna/diferença
  - Baixo hiato = pouca diferença
- Conceitos do domínio real **inspiram** os objetos de software, mas **não são a mesma coisa**
- O mais próximo que ficar do modelo de domínio, melhor
- Diminuição do hiato representacional é diretriz básica da tecnologia de objetos

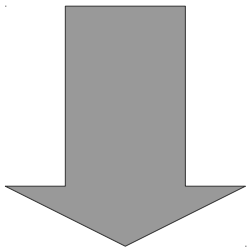
Baixo Hiato Representacional

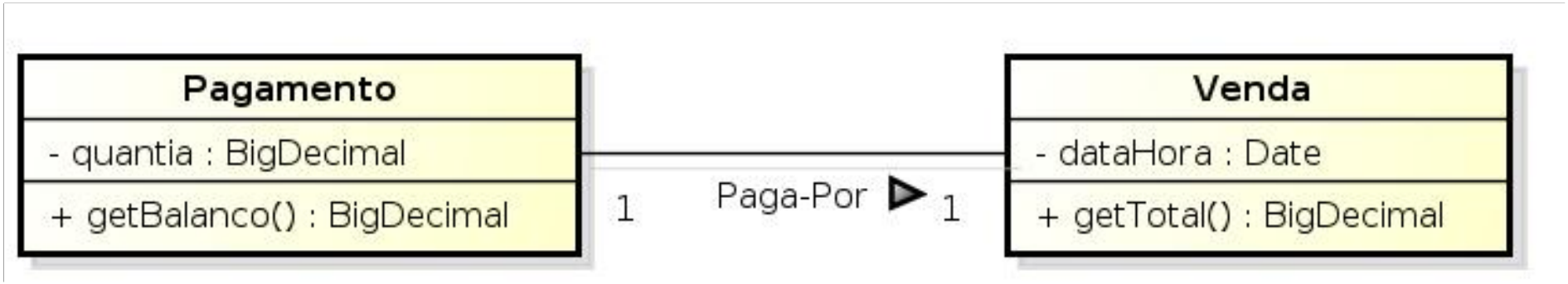
- Exemplo:

  Modelo inspira

  nomes e entidades

  do projeto





GRASP – Criador

- **Problema**: Quem cria o objeto A?
  - Uma das primeiras decisões de projeto
  - Início do ciclo de vida do objeto
- **Solução**: Delegar ao objeto B a criação de A se um ou mais das afirmativas for verdade:
  - B contém A, ou agrega de forma composta
  - B usa A de maneira muito próxima
  - B contém os dados iniciais de A

GRASP – Criador

- Exemplo
  - Sistema: Blog
  - Modelo de domínio
    - Blog contém posts
    - Post possuem título, corpo do texto, data e hora de publicação e comentários
    - Comentário é composto por autor e texto
  - **Quem cria um objeto 'Comentario' no sistema?**

GRASP – Especialista na Informação

- **Problema**: Qual o critério para atribuir responsabilidades aos objetos?
- **Solução**: Atribuir responsabilidade ao objeto que tenha informação necessária para satisfazê-la
  - Se um objeto precisa buscar toda a informação da sua responsabilidade em outras classes, pode haver algo errado
  - Abordagens mais apropriadas:
    - Manter nas responsabilidades do objeto apenas comportamentos relativos aos dados conhecidos
    - Delegar a outros objetos responsabilidades ligadas aos dados dos mesmos

GRASP – Especialista na Informação

- Promova o "**Diga, não Pergunte**" (*"**Tell, Don't Ask**"*)
  - *"Código procedural obtém informação e então toma decisões. Código OO diz aos objetos para fazerem coisas"* (Alec Sharp)
- Lei de Demétrio
  - "*Um método 'm' de um objeto 'O' deve somente invocar métodos dos seguintes tipos de objetos:*
    - *Do próprio 'O';*
    - *Parâmetros do método 'm';*
    - *Qualquer objeto instanciado dentro de 'm';*
    - *Componentes diretos de 'O';*
    - *Uma variável global, acessível por 'O', no escopo de 'm'.*"
-

GRASP – Especialista na Informação

- Implicações da Lei de Demétrio
  - Anti-exemplo

```
m(){
  // recupera outro objeto, e invoca função
  objetoQualquer.getOutroObjeto().facaFuncao();
}
```

  - Bom exemplo

```
m(){
  // delega invocação da função para objeto
    objetoQualquer.facaFuncao();
}
```

GRASP – Especialista na Informação

- Discussão:
  - Sistema: Comércio Eletrônico
  - Modelo de Domínio
    - Portal recebe pedidos, que contêm um ou mais produtos
    - Cupom de desconto podem ser vinculados ao pedido
    - A partir de determinado valor do pedido, o frete é grátis
    - Frete é calculado baseado no peso dos produtos e no CEP
  - **Quem armazena o frete?**

Separação Modelo-Visão

DIRETRIZ

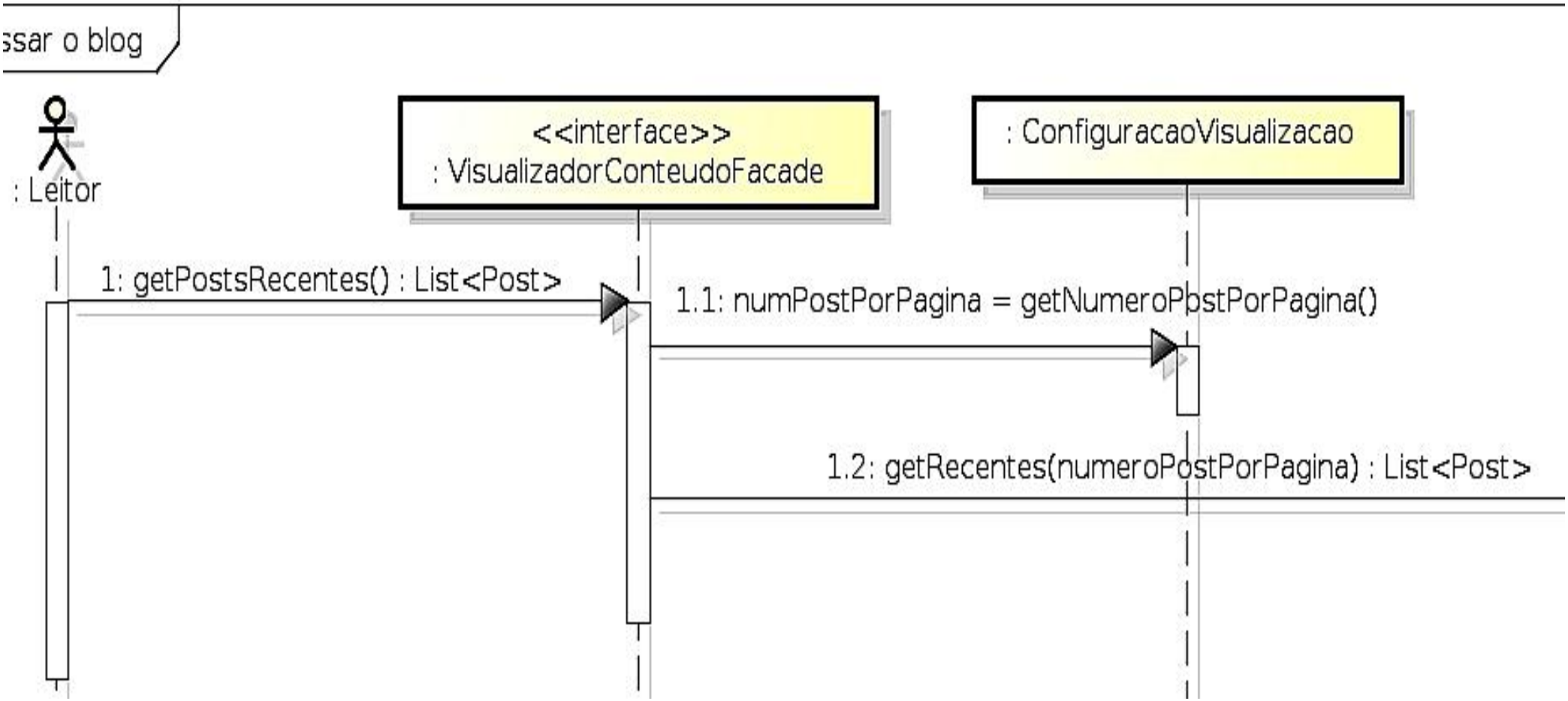
- Visão = elementos da Interface de Usuário (UI)
  - Componentes visuais ou diretamente relacionados
- Modelo = elementos do domínio ('negócio')
  - Componentes com as lógicas do modelo de domínio
- Não acoplar objetos de UI com objetos do domínio
- Não atribua a um objeto de UI responsabilidades do domínio (ex. Regras de negócio, funcionalidades)
- Objeto de UI apenas tratam os eventos, coletam dados e delegam solicitações à objetos do domínio

GRASP – Controlador

- **Problema**: Qual objeto recebe uma operação do sistema, além da camada de UI?
  - Se objetos UI não estão acoplados com objetos do domíno, quem está no meio do caminho?
- **Solução**: Atribuir a responsabilidade a um objeto que represente uma dessas escolhas:
  - Um subsistema, o sistema inteiro, ou um objeto “raiz”
  - Um caso de uso que contenha a operação

GRASP – Controlador

- Qual das duas escolhas foi feita no exemplo abaixo?



GRASP – Acoplamento Baixo

- **Problema**: Como reduzir o impacto de modificação?
- **Solução**: Atribuir responsabilidades favorecendo o acoplamento baixo
  - Grau de relacionamento com outros objetos
  - Avalie o acoplamento das possíveis soluções
  - Avalie o acoplamento de código legado ao realizar manutenção ou evolução

GRASP – Coesão Alta

- **Problema**: Como manter objetos focados, inteligíveis e gerenciáveis?
  - Por tabela, coesão apóia Acoplamento Baixo
- **Solução**: Atribuir responsabilidades de modo que a coesão permaneça alta.
  - Avalie a coesão das possíveis soluções
  - Avalie a coesão de código legado ao realizar manutenção ou evolução

GRASP – Coesão Alta

- Exemplo: **Qual alternativa parece ser a mais adequada?**