Projeto Detalhado de Software

GRASP
Indireção, Variações Protegidas, Revisão

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 25.
-

LABORATÓRIOS

•

Motivação

- **Por vezes, é importante evitar o acoplamento direto entre dois objetos.**
- Exemplos:
  - A $\rightarrow$ B, sendo B um pacote fornecido por fornecedor externo
  - A $\rightarrow$ B, sendo B uma implementação de acesso a um tipo específico de rede de comunicação ou protocolo
  - A $\rightarrow$ B, sendo B uma implementação de armazenamento persistente específica
  - Controlador para separar UI e Lógica de Negócio

GRASP - Indireção

- **Problema:** A quem atribuir responsabilidades para evitar acoplamento direto entre objetos?
  - Como continuar apoiando o acoplamento baixo, e potencializar o reuso?
- **Solução:** Usar um mediador entre os componentes, para evitar o acoplamento direto
  - A $\rightarrow$ Mediador $\rightarrow$ B, C, D...
    - A é cliente do mediador
    - B, C, D são serviços
  - O mediador cria uma indireção entre os componentes

GRASP – Indireção

- **Benefícios**
  - Protege o cliente do mediador de mudanças nos serviços
  - Promove o reuso dos serviços, através do mediador
  - Facilita evolução dos serviços, sem impacto nos clientes

"Os problemas na ciência da computação podem ser resolvidos por outro nível de indireção"

- David Wheeler

"... exceto o problema de muitas camadas de indireção"

- Kevlin Henney

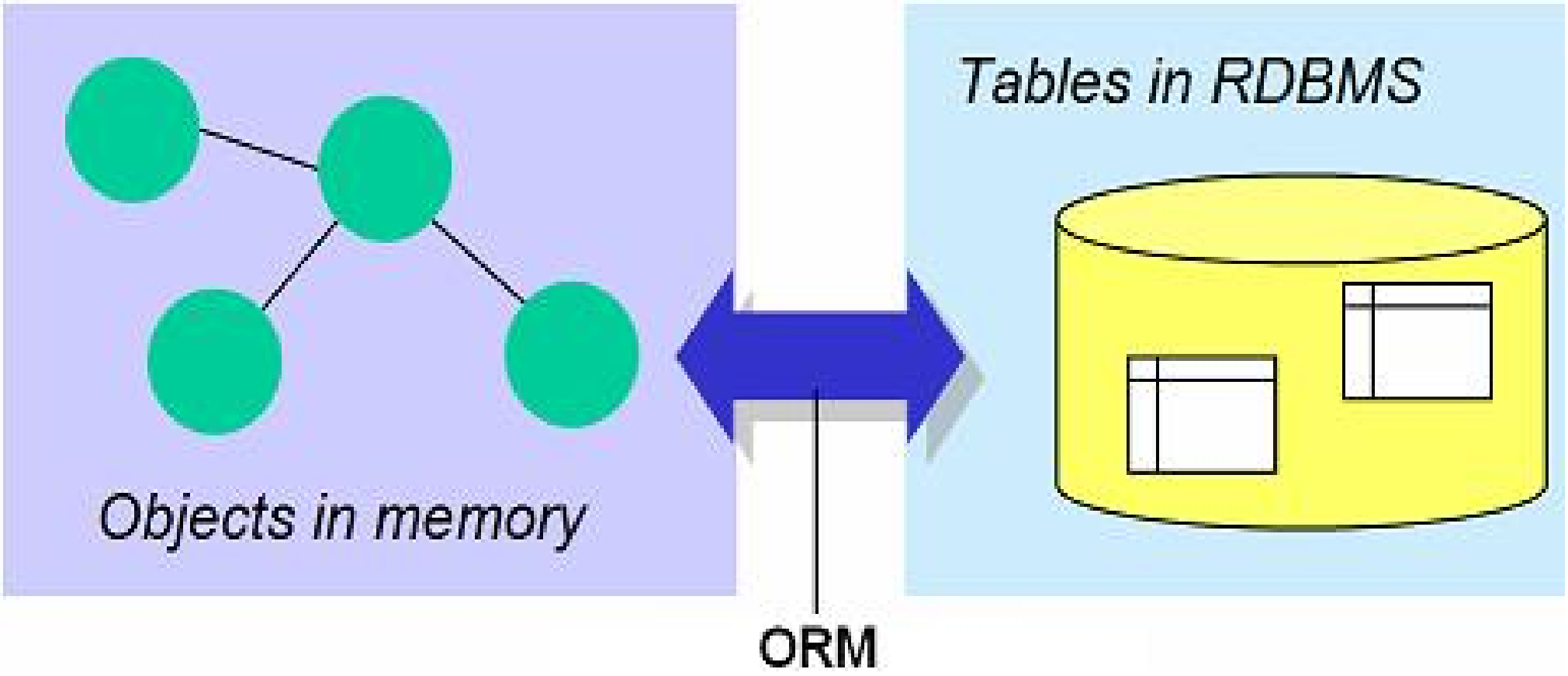
GRASP – Indireção

- **Exemplos na Computação**
  - Manipulação de valores usando endereço de memória
  - Uso de ponteiros para variáveis
  - Ligação dinâmica (mecanismo do polimorfismo)
  - URL para serviços na web
  - Nome de domínio para identificar máquinas
  - IPs para identificar nós na rede
-

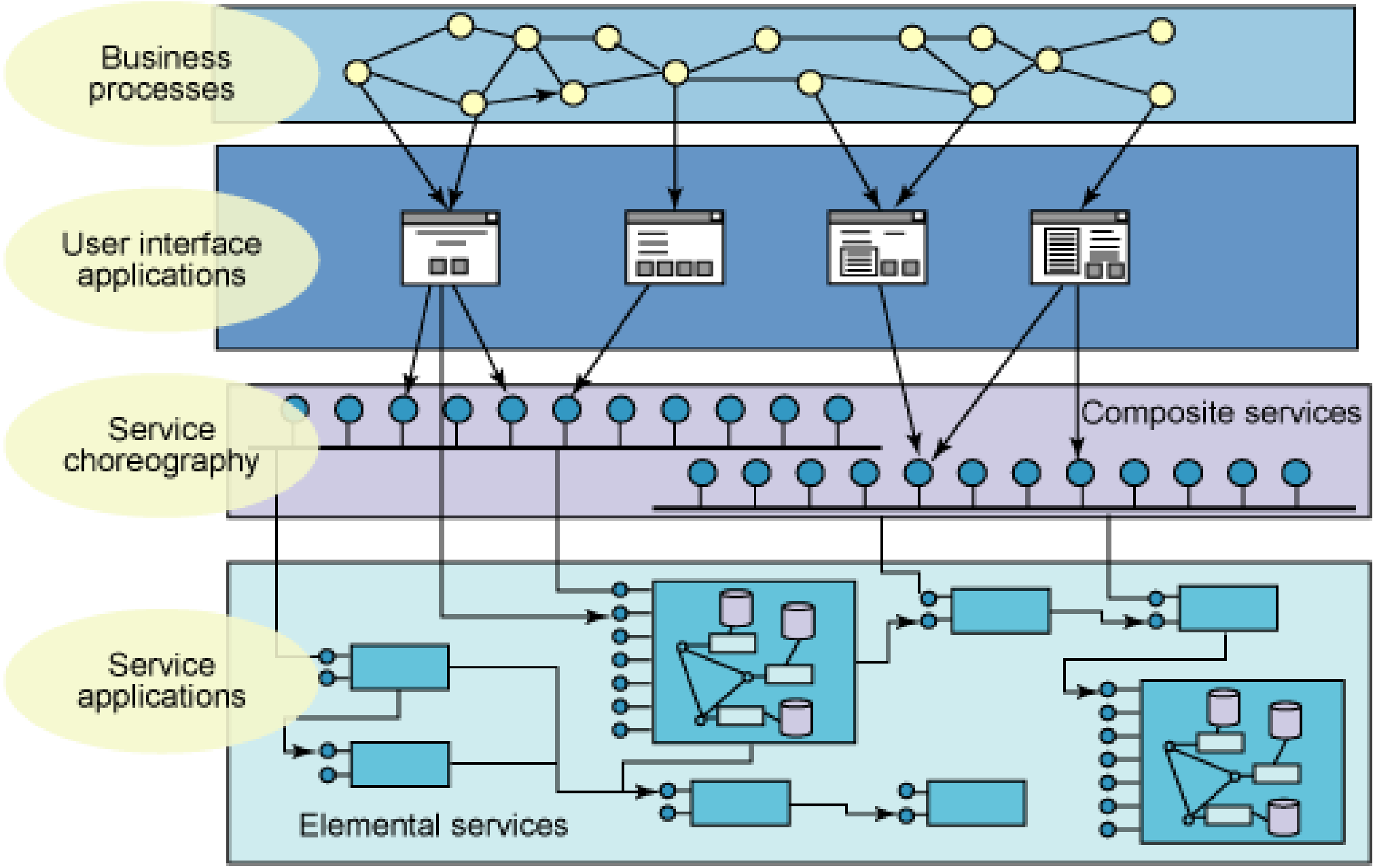
GRASP – Indireção

- **Exemplos na Engenharia de Software**
  - Componente para encapsular funções coesas
  - Componente para encapsular estrutura de dados
  - Controlador para ligar UI e domínio
  - Mapeamento Objeto-Relacional (ORM)
  - Vários Padrões GoF (Gang of Four), PoEAA (Fowler)
  - Service-Oriented Architecture (SOA)

GRASP – Indireção



GRASP – Indireção



Motivação – Variações Protegidas

- Sistemas são complexos para construir e para manter
  - Evoluir um sistema sempre será uma qualidade importante (mesmo que não dita)
  - A menos de sistemas descartáveis, todo sistema vai precisar evoluir
- Como proteger o sistema para suportar as variações futuras?
- Como lidar com variações ainda não solicitadas?

GRASP – Variações Protegidas

- **Problema:** como projetar objetos e subsistemas de modo que variações não causem impacto indesejados?

- **Solução:** identificar pontos de variação previsíveis, e atribuir responsabilidades criando interfaces em torno
  - Interface não é apenas a `interface` do Java
  - Tipos de Interfaces
    - Fronteira, API, Interface, Abstração, Protocolo, ...

GRASP – Variações Protegidas

- **Exemplo**:

  PDV com suporte para vários fornecedores de ECF

  – Solução: Interface + Polimorfismo

- **Variações protegidas**

  – Protocolos dos fornecedores

  – Inclusão de novo fornecedor de ECF

  – Nova regra da legislação

GRASP – Variações Protegidas

- Ponto de variação = ATUAL, NECESSÁRIO
- Ponto de evolução = FUTURO, PROVÁVEL
- Ponto de variação VERSUS Ponto de evolução
  - Cuidado com a especulação
  - Custo de trabalhar os Pontos de Evolução não pode ser demasiado
- Alta flexibilidade pode ser custosa hoje.
- Inflexibilidade pode ser custosa amanhã.

GRASP – Variações Protegidas

- Variações que aparecem recorrentemente nos projetos possuem soluções: são os Padrões!
- Exemplo: sistema de informação com banco de dados
  - Aplicação web + Banco de dados relacional
  - Variações a serem protegidas:
    - Modelo/esquema físico do banco de dados
    - Interface do usuário
    - Regras de negócio
    - Acesso simultâneo

GRASP – Revisão

- **Tipos de responsabilidade**
  - Fazer (algoritmos, rotinas, processamento)
    - *Ex. Calcular Frete*
  - Saber (dados)
    - *Ex. Endereço de entrega.*

GRASP – Revisão

- **Decomposição**
  - Representacional (ligados a conceitos do domínio)
    - *Ex.: Venda, Pedido.*
  - Comportamental (ligados a comportamento esperado)
    - *Ex.: Processamento do Pedido*
  - Invenção (comportamentos intrísecos ao domínio de desenvolvimento de software)
    - *Ex. Fila de Pedidos em Processamento.*

GRASP – Revisão

- **Processo de organização de responsabilidades**
  - Sempre orientado a requisitos e problemas
  - Para cada requisito, pode ser feita uma abordagem top-down ou bottom-up
    - Top-down: Partindo da interação dos atores até a lógica de negócios (mais fácil para identificar apenas elementos necessários)
    - Bottom-up: Partindo da lógica de negócios/dados, e chegando na interação dos atores (mais díficil, pode gerar elementos desnecessários)

GRASP – Revisão

- **Modelagem Visual**
  - Organiza os elementos e esboça as interfaces de componentes
  - Serve como comunicação entre equipe e stakeholders
  - Não é o fim, é apenas um meio de chegar no software rodando
- **Padrões didáticos**
  - Representam fundamentos básicos do design OO
  - Podem ser enxergados em outros padrões