http://www.agileandart.com/2010/07/16/ddd-introducao-a-domain-driven-design/

O que significa domain driven design?

projeto orientado a domínio

O que é DDD?

É um conjunto de práticas que tem por objetivo a construção de um software que expresse de forma bem clara o problema em questão.

Destaque as práticas de orientação a objetos?

* alinhamento do código com o negócio
* favorecer o reuso
* diminuir o acoplamento
* independência tecnologica

Quais as vantagens de usar DDD?

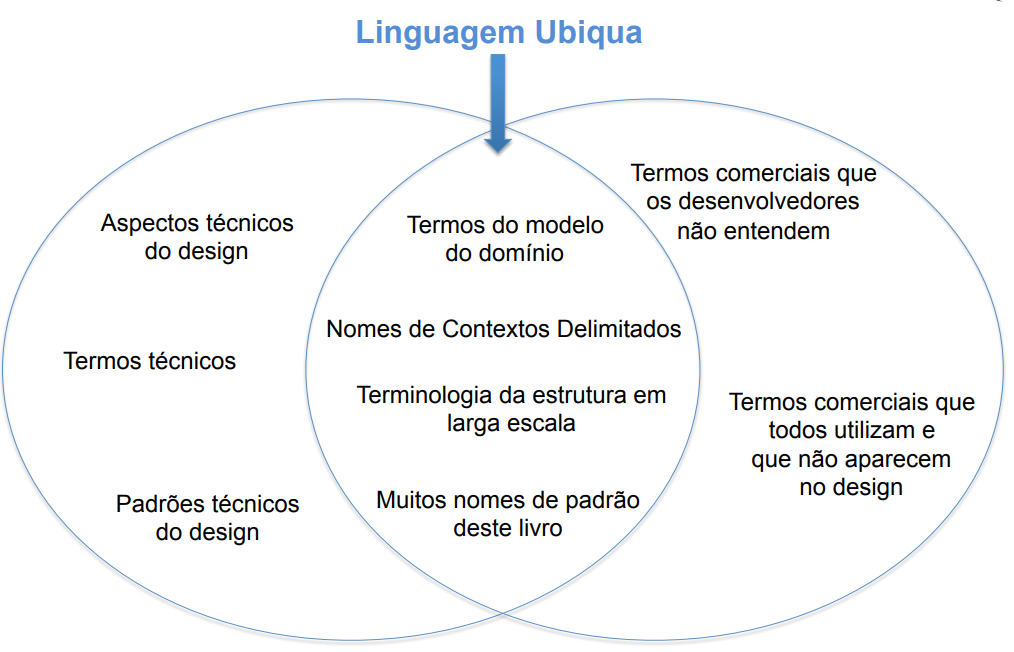
* O código fica menos acoplado e mais coeso
* O negócio é melhor compreendido por todos da equipe de desenvolvimento
* Facilita a comunicação
* Melhora a flexibilidade
* Enfatiza o domínio sobre a interface

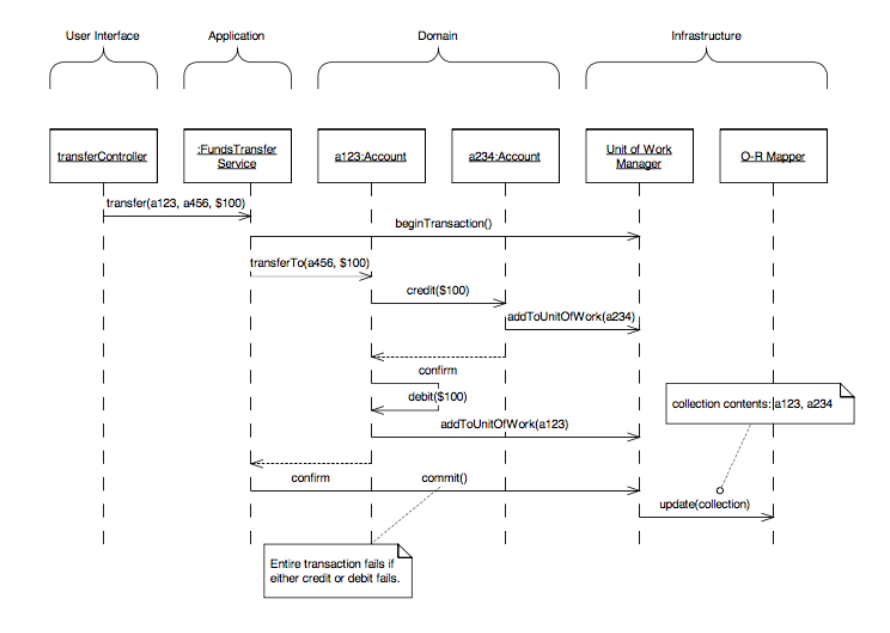
Quais são as desvantagens do DDD?

* Requer robusto expertise sobre domínio

O que é uma linguagem ubíqua?

linguagem comum, com termos bem definidos, que fazem parte do domínio do negócio e que são usadas por todas as pessoas que fazem parte do desenvolvimento do software





Mostre um exemplo de linguagem ubíqua?

Ex.:

Durante uma conversa com o cliente do sistema de cobrança, por exemplo, ele disser:

“Temos que **emitir** a **fatura** para o **cliente** antes da **data limite**”

Vamos ter no nosso código coisas do tipo:

* Uma classe para a entidade Cliente;
* Uma classe para a entidade Fatura;
* Algum serviço que tenha o método emitir;
* Algum atributo com o nome data limite.

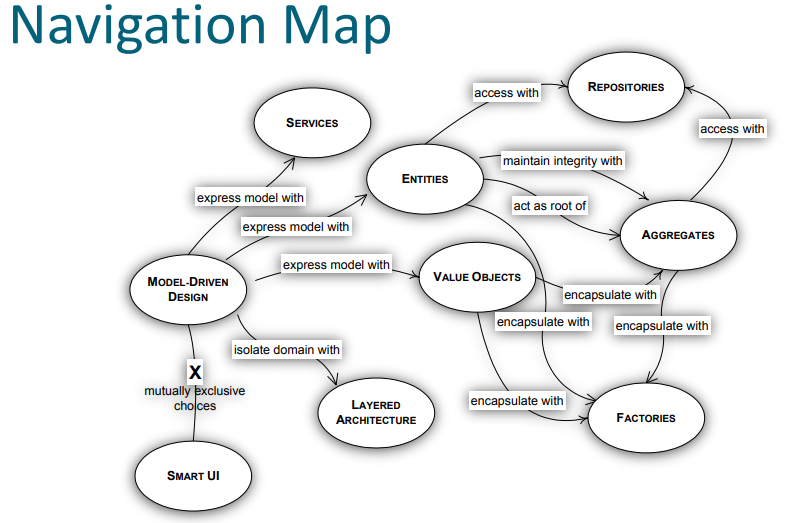
Qual o motivo de usar linguagem ubíqua?

A razão para usar linguagem ubíqua está no fato de criar um modelo de domínio por meio de Projeto Dirigido por Modelo (Model Driven Design - MDD).

Qual é a idéia do MDD?

* O seu modelo abstrato deve ser uma representação do seu domínio. Tudo que está no seu negócio deve aparecer no seu modelo.Só aparece no modelo aquilo que está no seu negócio
* Amarrar a implementação ao modelo
* Fornecer ferramentas para tornar eficiente
  + Uma rodada de engenharia de ferramentas de reversa
* Desenvolvedores e Modeladores estão acoplados por esta abordagem

Mapa de navegação do DDD?



Descreva as camadas do DDD.

User interface:

* Camada de apresentacao
* Mostrar informações
* Interpretar comando

Aplicação

* Camada fina, direciona comando de UI para a camada de dominio
* Não deveria conter regras de negócios ou conhecimento
* Nenhum “estado“ de negócio, poderia ter progresso de “estado“

Domínio

* Objetos de negócios, suas regras e seus estados
* A maior parte do livro foca neste ponto

Infraestrutura

* Capacidade técnica geral para suportar a camada de cima
* Envio de mensagens, perssitencia
* Suporta interação entre os padrões

Quais são as camadas do MDD?

Interface do usuário, aplicação, domínio, infra-estrutura

O que temos na camada de domínio?

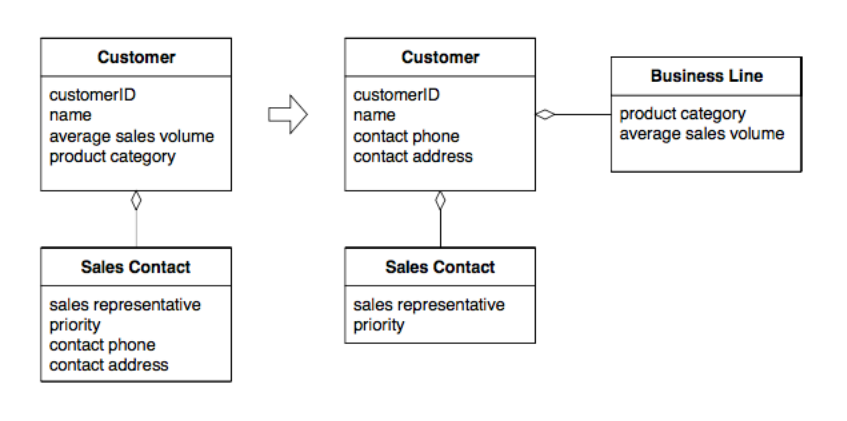
São padrões chamados de blocos de construção e serão utilizados para representar o modelo abstrato.

Quais são os blocos de construção da camada de domínio?

Entidade, Objetos de Valores, Agregados, Fábricas, Serviços, Repositórios, Módulos

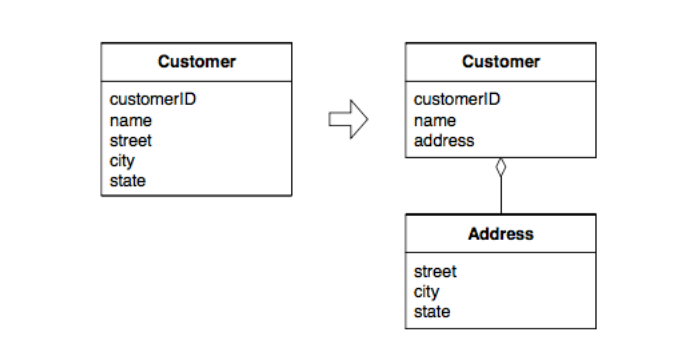
O que é Entidade?

* Tem identidade
  + Não o endereço do objeto
* O que é identidade do objeto?
  + Considere dois objetos, mesmo nome, mesma data de nascimento - separada pelas suas identidades.
  + Geralmente um identificador: número da conta



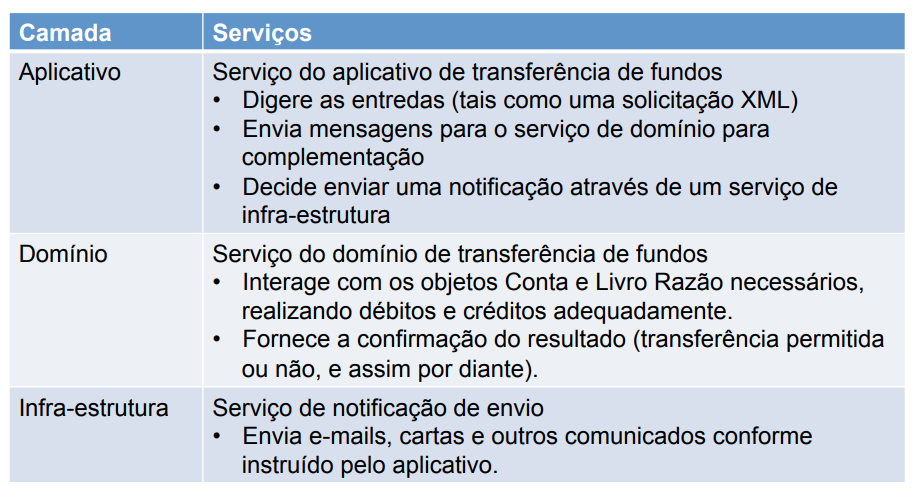
O que é Objetos de Valores?

* Nem todos os objetos são Entidades
  + Não podemos justificar o overhead de criar e rastrear identidades para todos os objetos.
* É recomendado que os objetos de valores sejam imutáveis.
  + classe Moeda para tratar dinheiro
  + classe Ponto para tratar desenhos
  + classe Endereco?



O que são Serviços?

* Alguns aspectos do domínio não mapeiam facilmente o objeto
* Um Serviço é algum comportamento, que é importante para o domínio, mas não pertence a uma Entidade ou a um Objeto de Valor.
* Exemplo: TransferirConta
* Encapsula conceito de domínio

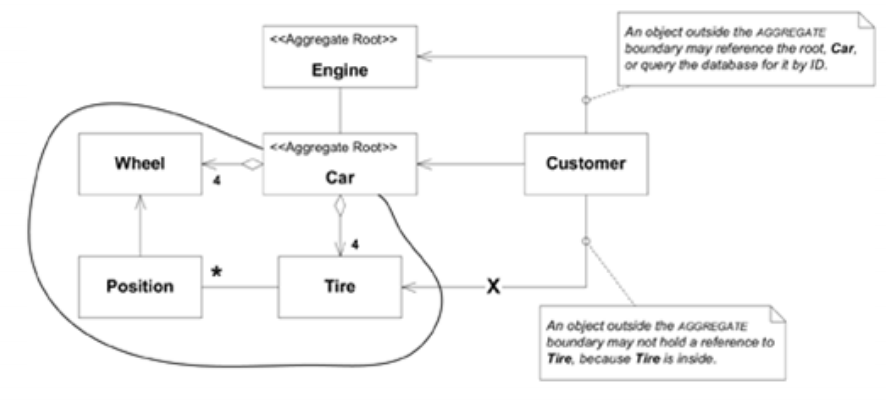


Quais são as características dos Serviços?

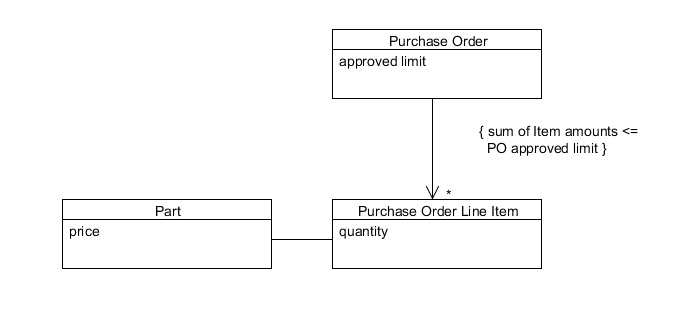
* A operação executada pelo Serviço refere ao conceito de domínio que não pertence a Entidade nem ao Objeto de Valor
* A operação executada refere-se a outros objetos do domínio
* A operação é stateless

O que são Agregados

* Um grupo associado de objetos que são considerados como unidades considerando mudança de dados
* Um agregado deve ter uma raiz
* A raiz é uma Entidade
* Objetos do lado de fora podem referenciar a raiz, mas não a outros membros Agregados.



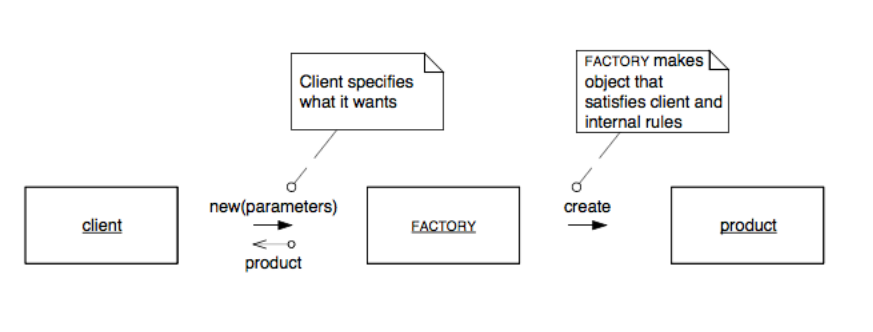
Exemplo

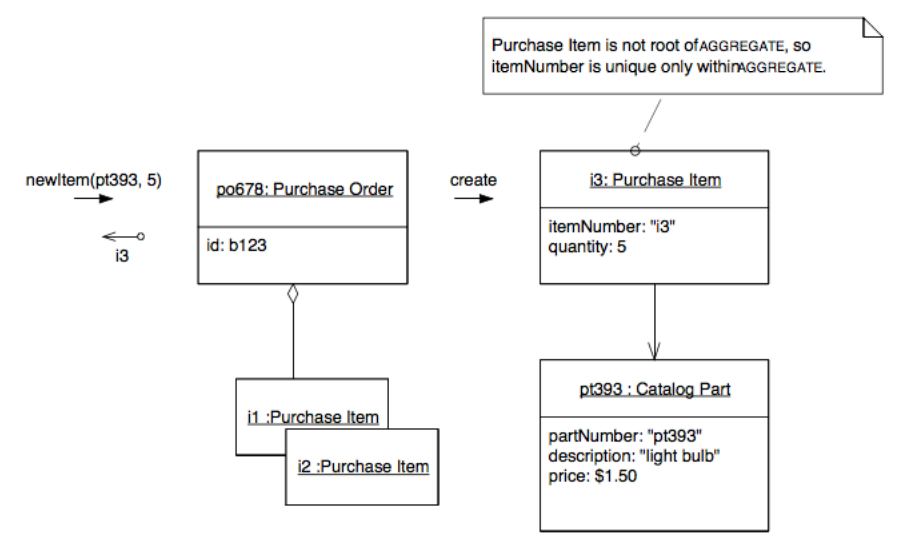


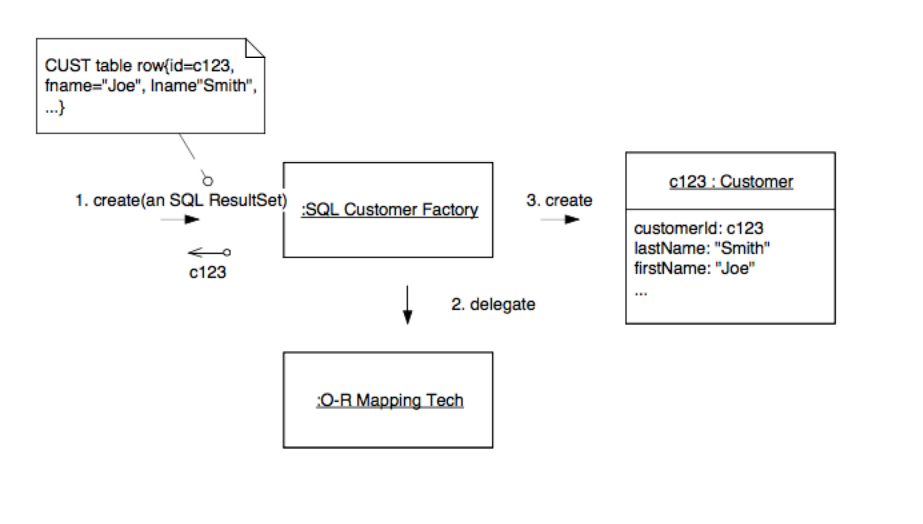
* Partes são usadas em muitas Purchase Orders (alta contenção)
* Poucas mudanças nas partes
* Mudanças nos preços da partes não propaga nas Purchase Orders

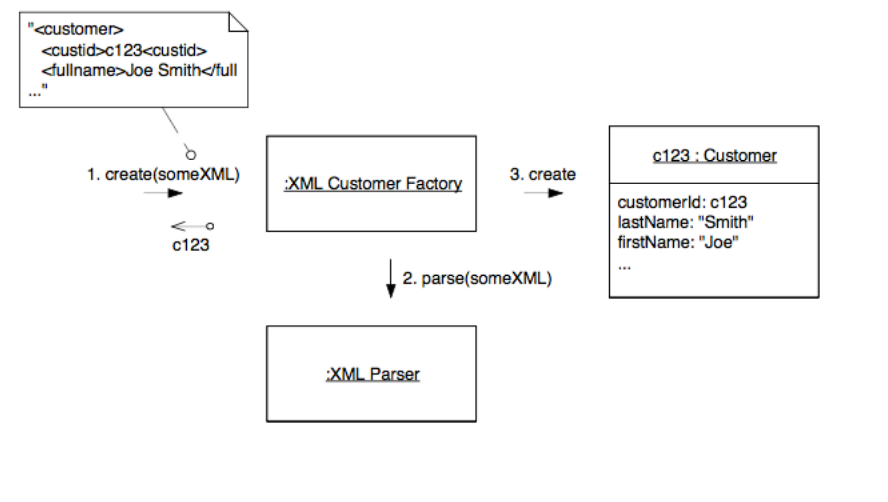
O que são Factories?

* Encapsula a informação necessária para a criação do objeto
  + Inclui lógica para todos criarem todos os membros e agregados
  + Permite reforçar a invariante durante a criação
  + Relacionar ao padrão GOF
    - Factory Method, Abstract Factory
  + Desenhar a interface Factory
    - Cada operação deve ser atômica
    - A Factory será acoplada aos seus argumentos



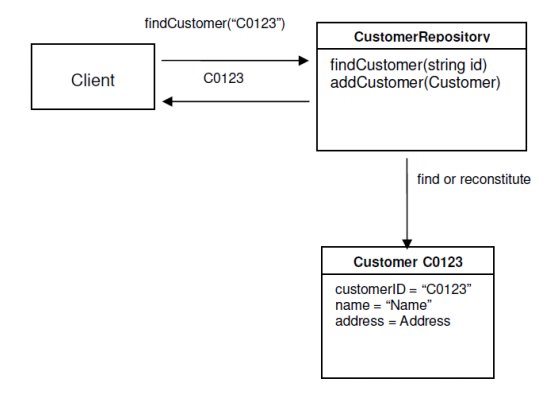


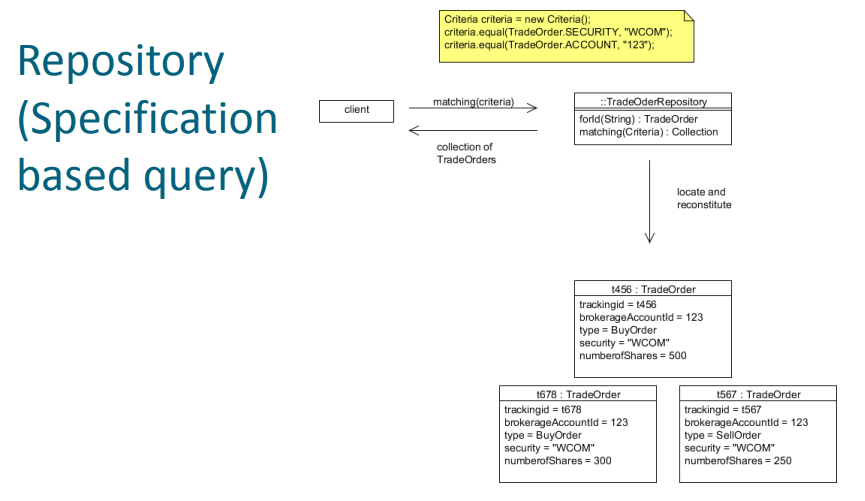


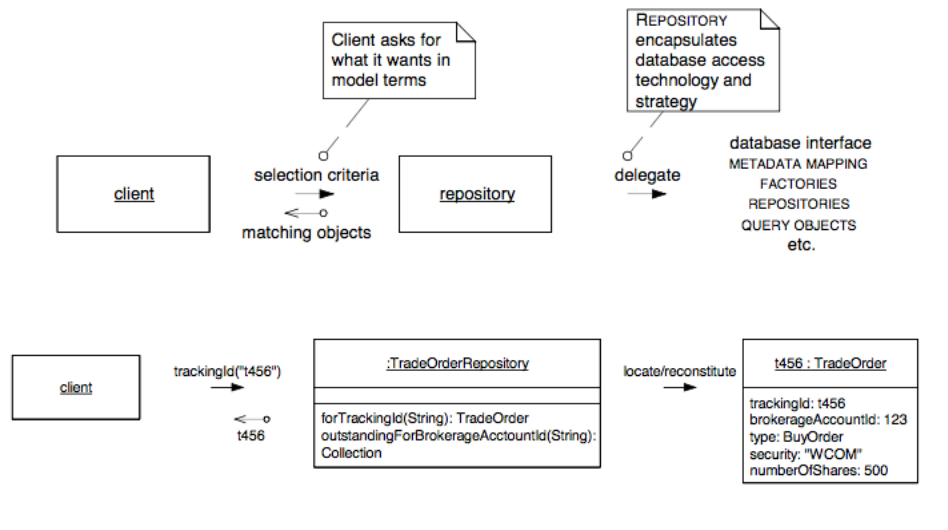


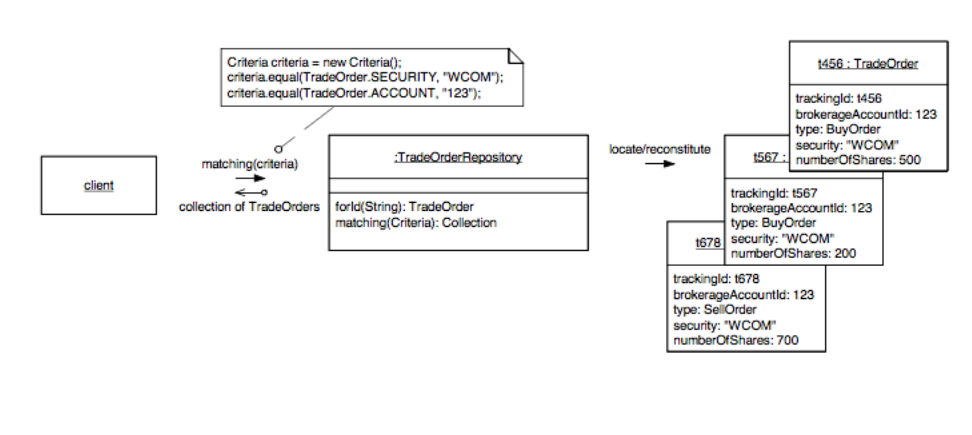
O que são Repositories?

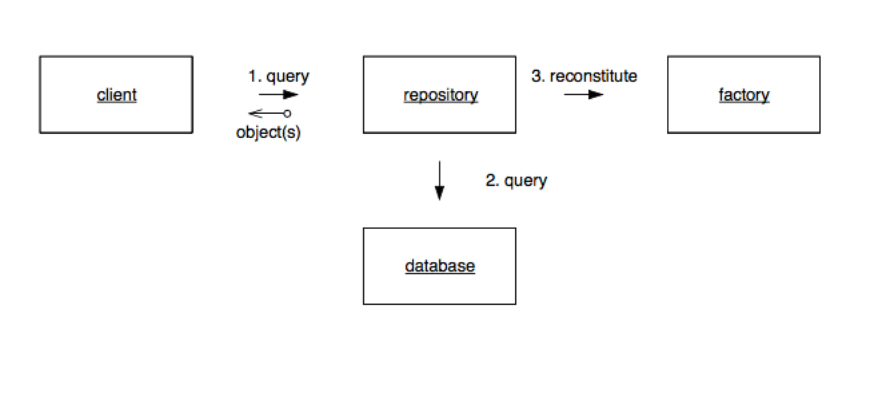
* Encapsula lógica para obter referências de objetos
* Fornece mecanismos para persistir/recuperar um objeto
  + Mantém código de persistência fora da camada de domínio
* A interface do Repositório deve ser dirigido pelo modelo de domínio
* A implementação do Repositório será ligado a infraestrutura

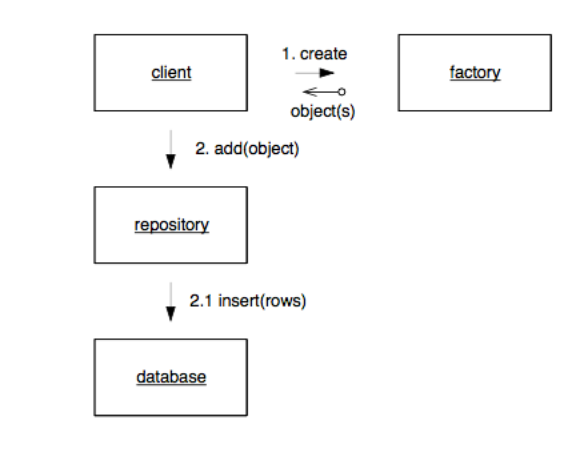


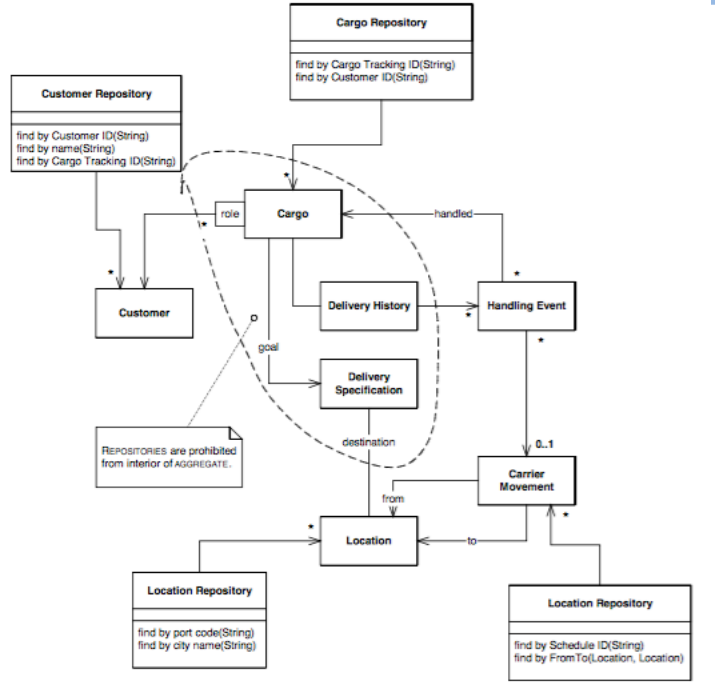


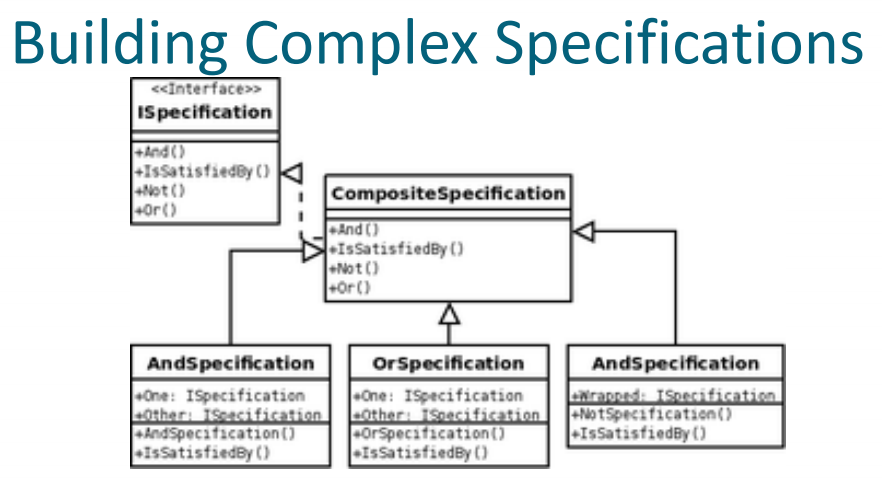












Dê um exemplo de módulos.

Exemplo:

módulo paciente tem:

* Ficha
* PrescriçãoMédica
* RegistroDeConsulta
* HistóricoDeCirurgia

Quando refatorar e ou melhorar? Na medida que se obtém compreensão dos conceitos importantes de domínio.

Quais padrões podem auxiliar a compreender mais profundamente o domínio?

* Interface de intenção revelada
* Funções sem efeitos colaterais
* Asserções

Como é o padrão Interface de intenção revelada?

Usar nome de métodos ou classes que dizem exatamente “o que” elas fazem e não como.

O que são funções sem efeitos colaterais?

Deixar o código com o maior número de métodos que não alterem o estado interno dos objetos, concentrando esse tipo de operação (alteração do estado) em Comandos;

Objetivo do DDD? Criar um sistema dirigido pelas regras de negócio do domínio.

Quando o DDD se torna importante e complicado? Quando começamos a lidar com sistemas complexos.

Como dividir o software? Como deve ser interação entre sistemas que interagem entre si?

Como fazer a comunicação entre os times de desenvolvimento?

Temos padrões que ajudam a dividir nosso software em várias partes, que chamamos de contextos.

Quais padrões ajudam estabelecer a relação entre vários times?

Mapas de contextos, contextos delimitados (bounded context), produtor-consumidor, conformista, núcleo compartilhado, camada anti-corrupção, caminhos separados, serviço aberto de funcionalidades e linguagem publicadas.

O que é a destilação do domínio? Quando não temos mais nada para tirar do que chamamos de núcleo do domínio (core domain) .

O que é a sentença da visão do domínio?

É um texto descrevendo o núcleo do domínio.

Dê um exemplo de sentença de dominio.

*“O modelo representa compras feitas por clientes numa loja. O cliente pode passear pela loja e visualizar produtos, que estão divididos em categorias. O cliente coloca os produtos que deseja num carrinho de compras e depois passa no caixa para efetuar o pagamento, que pode ser feito em dinheiro ou cartão. Após o pagamento, os produtos são oficialmente retirados do estoque da loja.”*

Dê um exemplo de uma sentença que não faz parte do domínio.

*“Todos os produtos são identificados por códigos de barra. Os códigos são lidos no caixa por um terminal e os preços são mostrados na tela, para o cliente, quando o código é lido. Os dados de compra são enviados através da rede sem fio e cadastrados num banco de dados, que deve ter 99,9% de disponibilidade.”*

.