MODELANDO DOMÍNIOS – PRINCÍPIOS E PRÁTICAS

Prof. Msc. Rafael Cruz

MICROSOFT MOST VALUABLE PROFISSIONAL

DOMAIN DRIVEN DESIGN

- Todo o Software possui alguns atributos em comum:
    - Quantidade de dados
    - Performance
    - Complexidade de Regra de Negócio
    - Complexidade Técnica

DOMAIN DRIVEN DESIGN

- O objetivo do DDD é produzir sistemas fortemente orientado a negócios que sejam estáveis e fáceis de manter

  - DDD foca em simplificar o problema, extraí-lo e deixá-lo CLARO no código.
  - DDD deve ser a solução no seu projeto e não o problema!
  - O DDD é uma abordagem de modelagem de software com objetivo de facilitar a implementação de processos de negócios
  - Domain Driven Design como o nome já diz é sobre o design guiado pelo domínio. Uma modelagem de software focada em resolver problemas na complexidade do negócio.

PRINCIPAIS CONCEITOS - Ubiquitous Language

- É uma linguagem que deve ser comum entre o "Cliente" e o "Desenvolvedor".

- Seu código deve expressar as terminologias do cliente, para que os desenvolvedores e o cliente falem a mesma língua.

PRINCIPAIS CONCEITOS – Bounded Context

- Criar delimitações específicas entre cada parte do sistema.
- Uma classe Produto não tem o mesmo sentido num contexto de "Suporte" do que num contexto de "Venda".
- Essa "Separação" de sentido entre contextos DEVE SER CLARA!

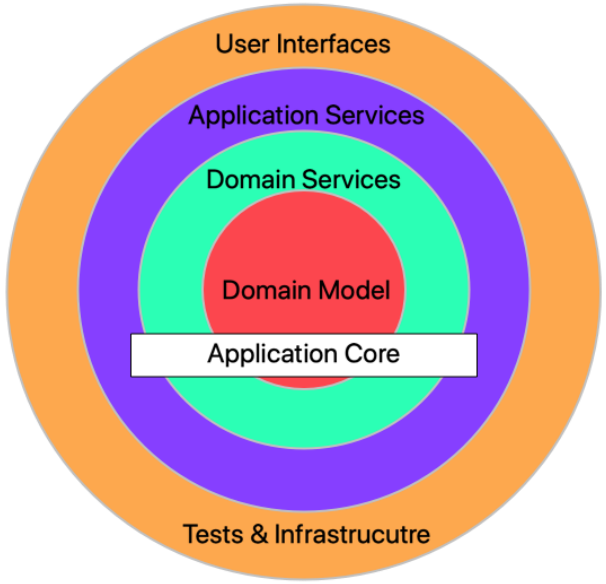
DDD Is Not Only About Writing Code

Os desenvolvedores devem estar em constante contato com o negócio e evoluírem constantemente no entendimento sobre o domínio.

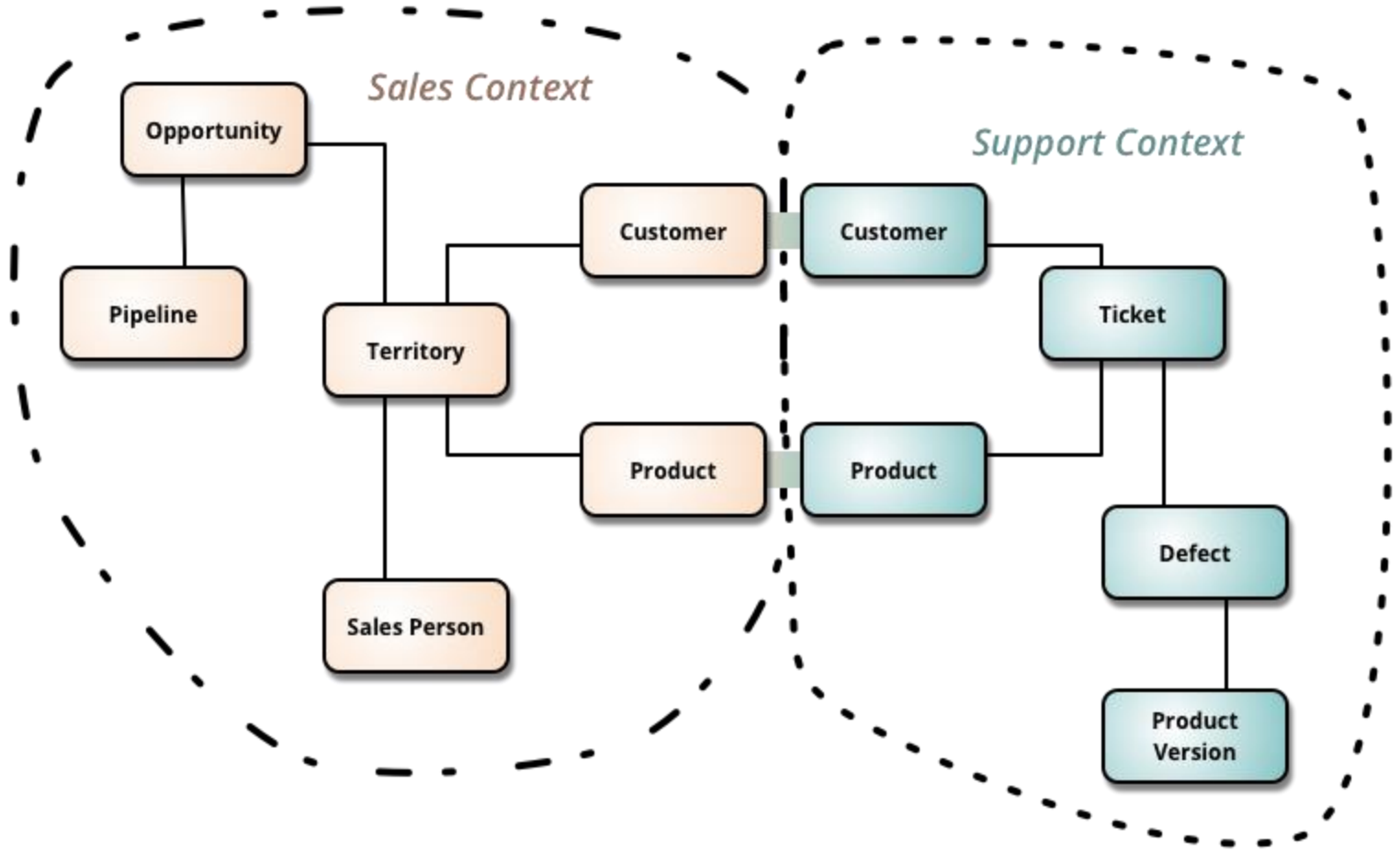
Todos os desenvolvedores devem estar cientes do ponto de vista do Domain Expert e do que estão cumprindo.

Application Architecture

- As regras de negócios estarão nos objetos centrais, em especial, as Entity e Value Object
- Esses objetos não devem ter CONHECIMENTO NENHUM, de como vão ser persistidos, construídos ou mapeados no banco de dados.
- O banco de dados estará sempre na camada Repository, se você vai utilizar uma ORM ou efetuar as query diretamente, não importa! Mas as interações com o banco ocorrerão por lá.



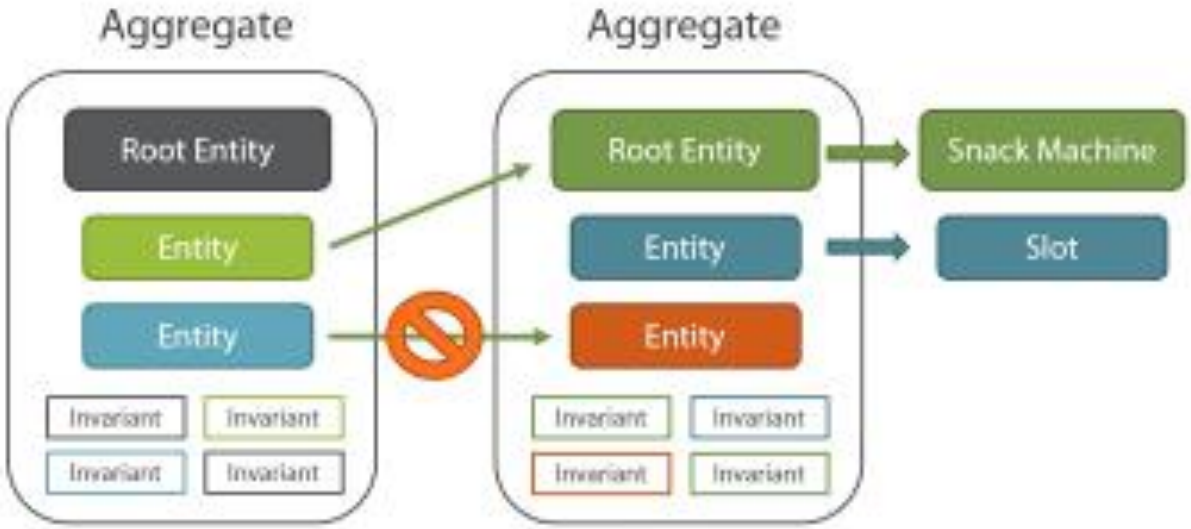
Aggregates


Sales Context
Support Context
Opportunity
Pipeline
Territory
Sales Person
Customer
Product
Customer
Product
Ticket
Defect
Product Version

Aggregates

Aggregate é um Design Pattern dentro do DDD que busca simplificar as responsabilidades da Entidade. Ele irá conter uma ou mais Entity e será responsável por efetuar as validações e aplicar regras.

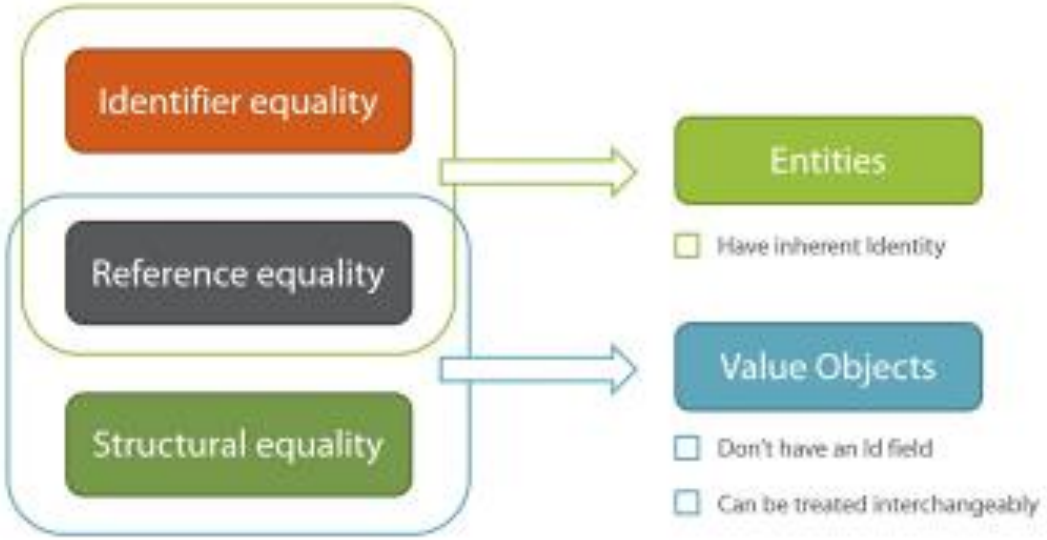
Um Aggregate deve sempre possuir uma Entidade "Raíz", que representa o Domínio ao qual aquela regra pertence



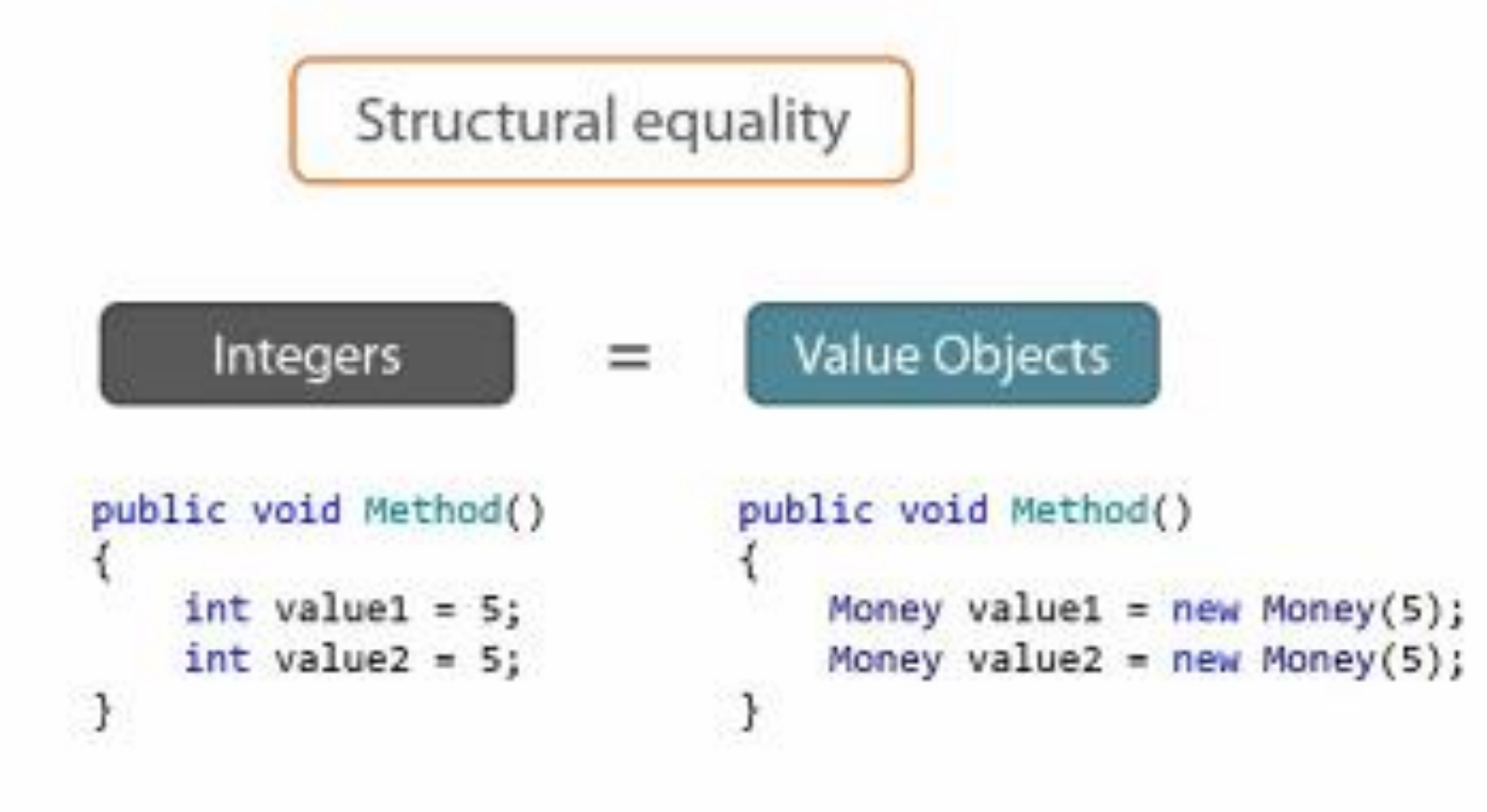
Entity vs Value Objects

Entity possuí uma identidade única, cada uma possuí produtos diferentes, quantidades diferentes, valores diferentes e são únicas, não podemos simplesmente trocar uma pela outra.

Value Objects não possuem uma identidade, eles podem ser trocados! Pense numa classe "Money" por exemplo, ela não tem uma identidade própria, ela pode ser trocada por qualquer outra classe que tenha a mesma composição



Dê preferencia à Value Object



Repositories

Repositório é um padrão que visa encapsular toda comunicação com o banco de dados.

O foco desse padrão, é abstrair a lógica de capturação dos dados e retornar as entidades modelo como se elas já estivessem alí, prontas para serem capturadas.