Entendendo as diferenças entre BDD, TDD e DDD

Carolina Santana Louzada

Analista QA - Venturus

Objetivo do curso

Reforçar comparativamente conceitos que podem ser confundidos, mas amplamente utilizados no desenvolvimento de software como BDD, TDD e DDD.

Pré-requisitos

- Fundamentos de qualidade de software
- Ciclo de Desenvolvimento de Software e Metodologias Ágeis

Percurso

Aula 1

Afinal, qual a diferença entre BDD, TDD e DDD?

Dúvidas durante o curso?

> Fórum do curso

> Comunidade online (Discord)

SCAN ME

Aula 1

Afinal, qual a diferença entre BDD, TDD e DDD?

// Entendendo as diferenças entre BDD, TDD e DDD

Objetivos

1. Revisar o conceito de TDD
2. Revisar o conceito de BDD
3. Revisar o conceito de DDD
4. Entender comparativamente os conceitos revisados

Aula 1 . Etapa 1

O que é TDD?

// Entendendo as diferenças entre BDD, TDD e DDD

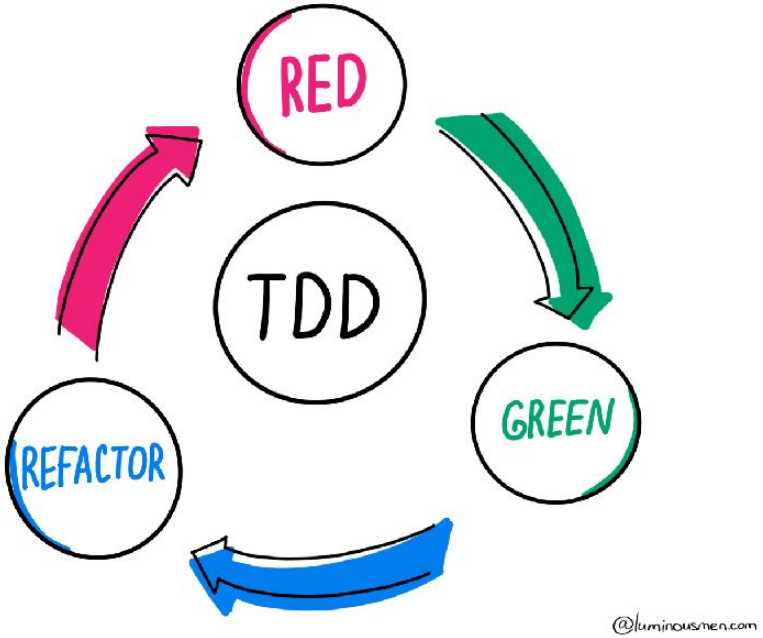
Surgimento do TDD

- Desenvolvido por *Kent Beck* na mesma época do desenvolvimento dos modelos ágeis, em específico do XP. *(Extreme Programming)*
- Inversão da ordem tradicional de desenvolvimento, na qual os casos de testes automatizados são criados antes da efetiva implementação.
- **Qual a ideia por trás ?** Desenvolvedores participam da captura e análise dos requisitos, moldando dessa forma a construção do código-fonte.

Benefícios

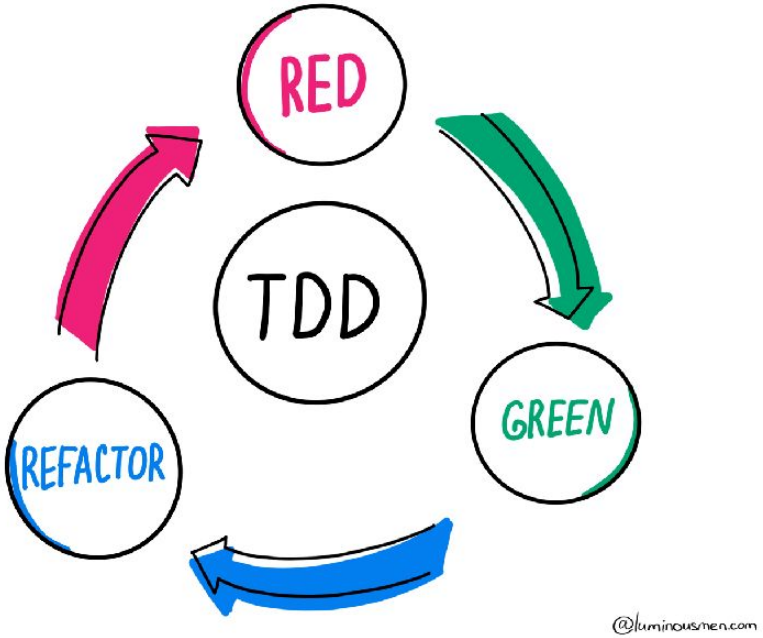
- ➔ Auxílio para emergência de designs e suas implementações
- ➔ Auxílio para realizar alterações motivadas por mudanças de requisitos
- ➔ Apoio para exploração de bibliotecas providas por terceiros
- ➔ Apoio para promover alterações estéticas em códigos-fonte

Processo simplificado do TDD



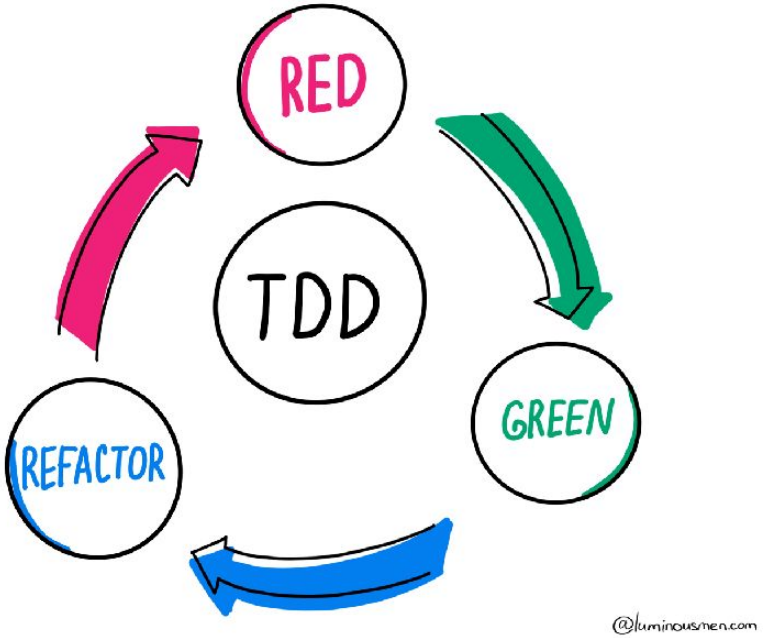
★ Estratégia *bottom-up* na qual o software é elaborado o quanto antes a partir de partes elementares.

Processo simplificado do TDD



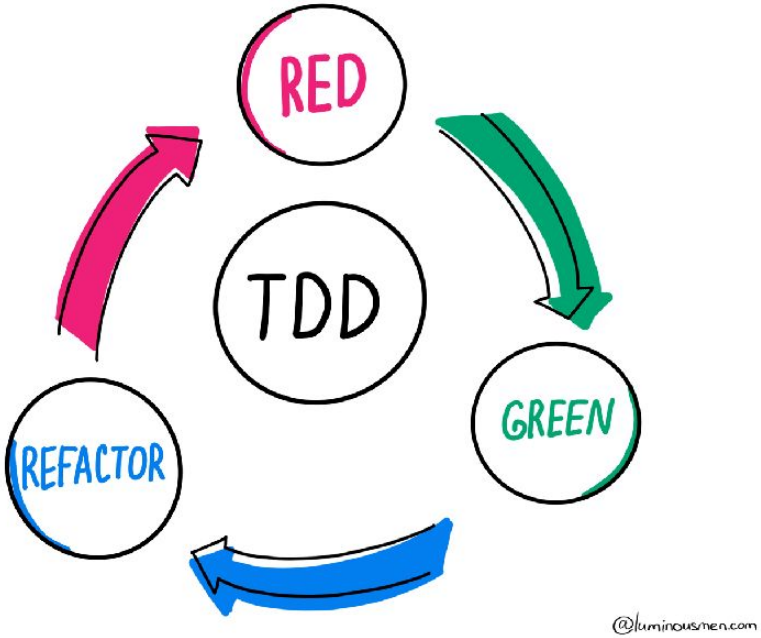
★ RED : testes unitários são elaborados para estimular o pensamento do design da aplicação, de forma que os testes ainda possam falhar inicialmente.

Processo simplificado do TDD



★ GREEN: O código é construído e pensado para que passe no teste desenvolvido “sem pensar demais”.

Processo simplificado do TDD



★ Refactor: Momento de limpar e melhorar o código

Test Driven Development: TDD Simples e Prático (devmedia.com.br)

Aula 1 . Etapa 2

O que é BDD?

// Entendendo as diferenças entre BDD, TDD e DDD

Surgimento

- ★ Conceituado por Dan North em meados dos anos 2000
- ★ A questão estava indo mais além do processo da construção e implementação do software em si, mas também do próprio entendimento do que o cliente deseja, algo que até hoje é um dos pontos mais complexos: a elicitação dos requisitos!
- ★ O TDD somente não era suficiente para abrir a porta do entendimento!

Introducing BDD - Dan North & Associates Ltd

Benefícios

- ★ Compartilhamento de conhecimento
- ★ Documentação dinâmica
- ★ Interação entre membros da equipe
- ★ Visão do todo

A história do usuário

- O ponto focal no entendimento de uma funcionalidade requerida pelo usuário, contudo isso por si só não garante que os cenários associados sejam condizentes com o que o cliente deseja ver no software na prática.
  - Quem é o usuário?
  - O que ele quer atingir?
  - Qual o benefício disso?

A história do usuário - Um exemplo

Como gerente da equipe

Eu quero criar equipes formadas por funcionários cadastrados de diversas áreas

Para que eu possa gerenciá-los

Critérios de aceite

- Tomando como o exemplo anterior, como a equipe de desenvolvimento conseguiria compreender completamente aquilo que o cliente deseja?
  - Critérios de Aceite
- São os requisitos e regras principais que norteiam a implementação do software de forma que o cliente considere aquela funcionalidade como entregue.

E como isso se encaixa com o BDD?

1º Se torna uma prática de desenvolvimento no sentido que o desenvolvimento e os testes serão considerados a partir da visão do comportamento do sistema. O que importa é o “COMO” e ter exemplos concretos de como os fluxos vão funcionar!

2º Esses exemplos ou cenários são transformados nos critérios de aceite e portanto são oficializados a partir de um documento acordado e entendido por todos a partir de uma especificação da linguagem.

E como isso se encaixa com o BDD?

3º A especificação é automatizada para verificar se o comportamento é atendido.

Mas e o Cucumber? E o Gherkin?

- ★ Cucumber e Gherkin =! BDD
- ★ **Cucumber** é uma ferramenta que lê a documentação gerada e a transforma em testes automatizados através da ideia dos cenários e exemplos.
- ★ **Gherkin** é uma linguagem ou conjunto de regras gramaticais que estrutura o texto das especificações de forma que a ferramenta possa entendê-las e gerar os passos para a automação.

Mas e o Cucumber? E o Gherkin?

★ **Gherkin** é uma linguagem ou conjunto de regras gramaticais que estrutura o texto das especificações de forma que a ferramenta possa entendê-las e gerar os passos para a automação.

Mas e o Cucumber? E o Gherkin?

Funcionalidades: Controlar o Tráfego Aéreo

Como um controlador de tráfego aéreo
Eu quero obter informações meteorológicas e climáticas
Para utilizar estas informações em outras atividades do Sistema

Cenário 1: Ajustar Altímetro
Dado que estou em vôo nivelado
E vou iniciar o procedimento de sua aproximação para pouso
E defino "altitude"
E defino "pressão atmosférica"
E defino "temperatura"
Quando "reunir" as informações altitude + pressão atmosférica+ temperatura
Então preciso ver "o valor a ser ajustado no altímetro"

Cenário 2: Identificar Altitude
Dado que temos uma atmosfera padrão
E defino "pressão padrão"
E defino "nível médio do mar" (QFF)
Quando "reunir" as informações pressão padrão + QFF
Então preciso ver "a distância vertical que separa

Modelo de Desenvolvimento de Objetos de Aprendizagem Baseado em Metodologias Ágeis - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net

Aula 1 . Etapa 3

O que é DDD?

// Entendendo as diferenças entre BDD, TDD e DDD

Surgimento

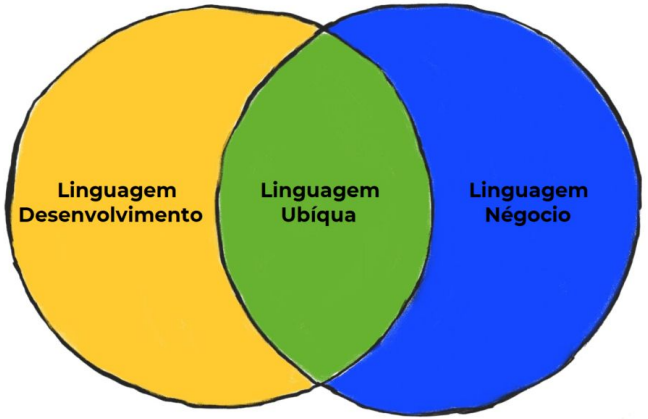
- ★ Termo surgiu a partir do livro escrito por Eric Evans em 2003
- ★ Baseia toda uma estrutura para tomada de decisões, indo do design até o desenvolvimento do software em si, norteando todo o processo com a ideia de que o software está relacionado a um contexto, conhecimento, processos e problemas a serem resolvidos de forma integrada ao cliente/usuário.
  - ○ Isso que chamamos de domínio!

Ferramentas de design

- Modelagem de software -> Design estratégico
  - princípios e padrões para dividir um problema complexo em blocos claros e com responsabilidades específicas, formando uma ‘topologia’ de alto nível
- Implementação -> Design tático
  - Padrões de abstração de componentes em médio e baixo nível, tendo um refinamento da estratégia adotada a partir do código.

Design estratégico

- ➔ Bound Contexts
- ➔ Linguagem Úbiqua
- ➔ Context Maps



James Shore.The Art of Agile Development

Colaboração da equipe e especialistas do domínio

Quem entende do domínio?

- Domain Expert
  - Especialista que sabe as necessidades do negócio, apoiando portanto o time de desenvolvimento para modelar o domínio, descrevendo o que sistema deve fazer e seu objetivo

Como todo mundo se entende?

- ➔ Desenvolvedores dificilmente entenderão profundamente sobre a necessidade do negócio ou das terminologias usadas, mas entendem da parte prática do código.
- ➔ Por sua vez, os especialistas do domínio dificilmente serão desenvolvedores e não entenderão muitas terminologias da área técnica.
- ➔ Como fazer todo mundo da equipe e o software se entenderem?

Linguagem Ubíqua

Como todo mundo se entende?

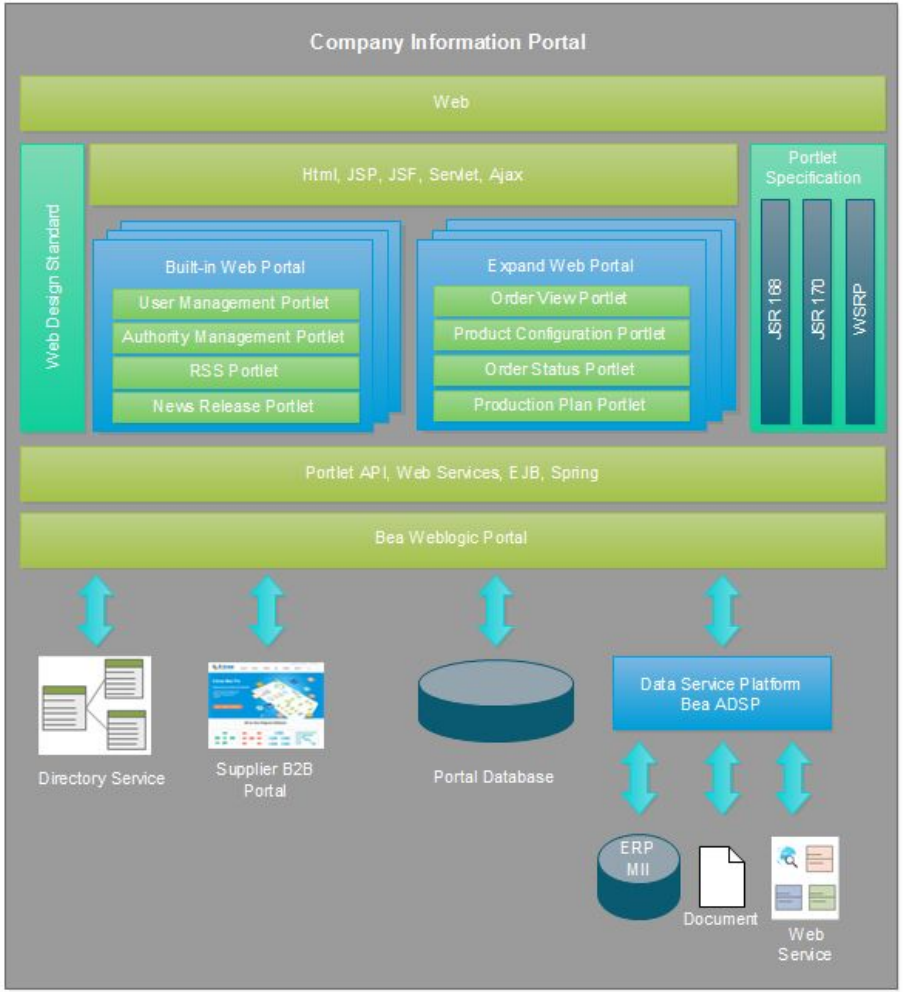
- ➔ A linguagem comum é importante para evitar ambiguidades e erros gerados por mal entendimentos de termos.
- ➔ A integração da equipe gera feedbacks mais confiáveis e fortalece o entendimento do software.

Bounded Context

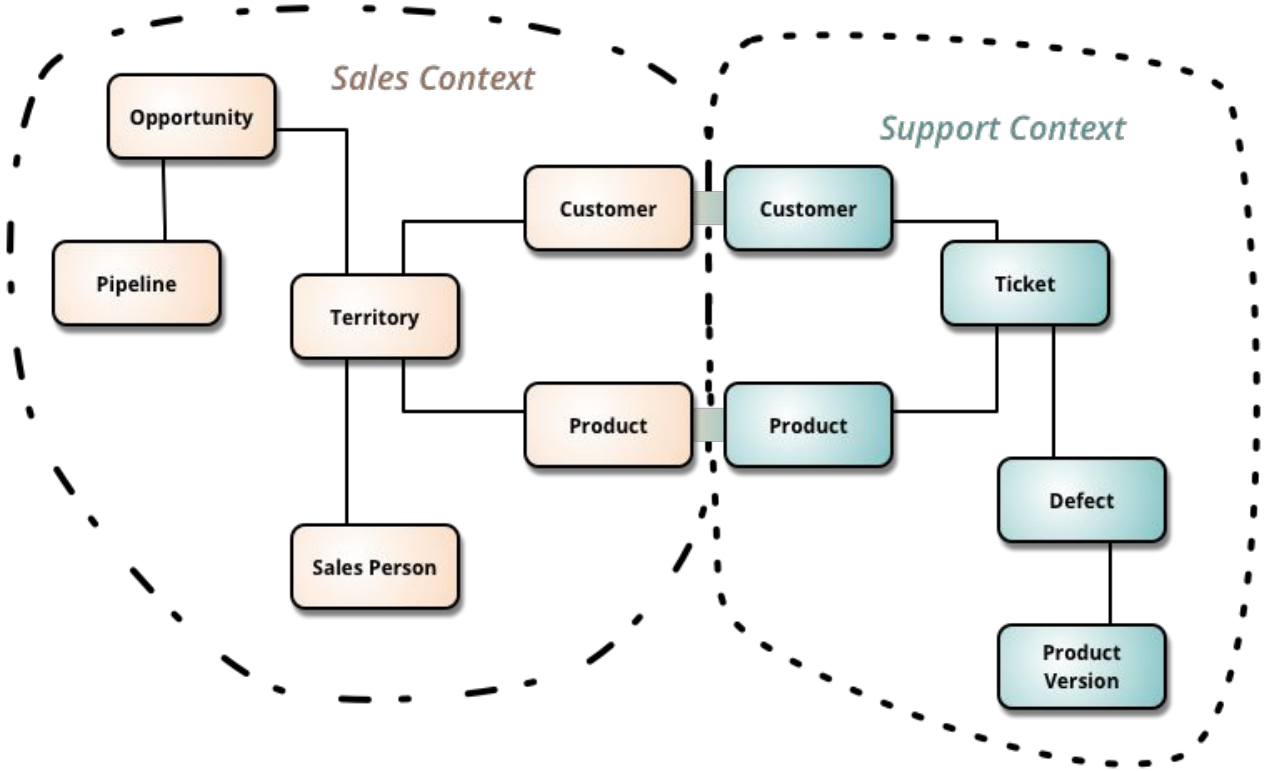
- ➔ Domínios complexos dificilmente levarão à construção de um modelo de pensamento unificado .
- ➔ A ideia é “dividir para conquistar”
  - ◆ O domínio deve ser dividido em contextos menores que são inter-relacionados, mas são claramente divididos.

Context Map

- ➔ Visão global do software a partir de um documento ou esboço que facilite o entendimento dos contexto da aplicação.
- ➔ É de alto nível e não precisa ser algo complexo a níveis de diagramas arquiteturais, mas deve abranger as relações entre os contextos.

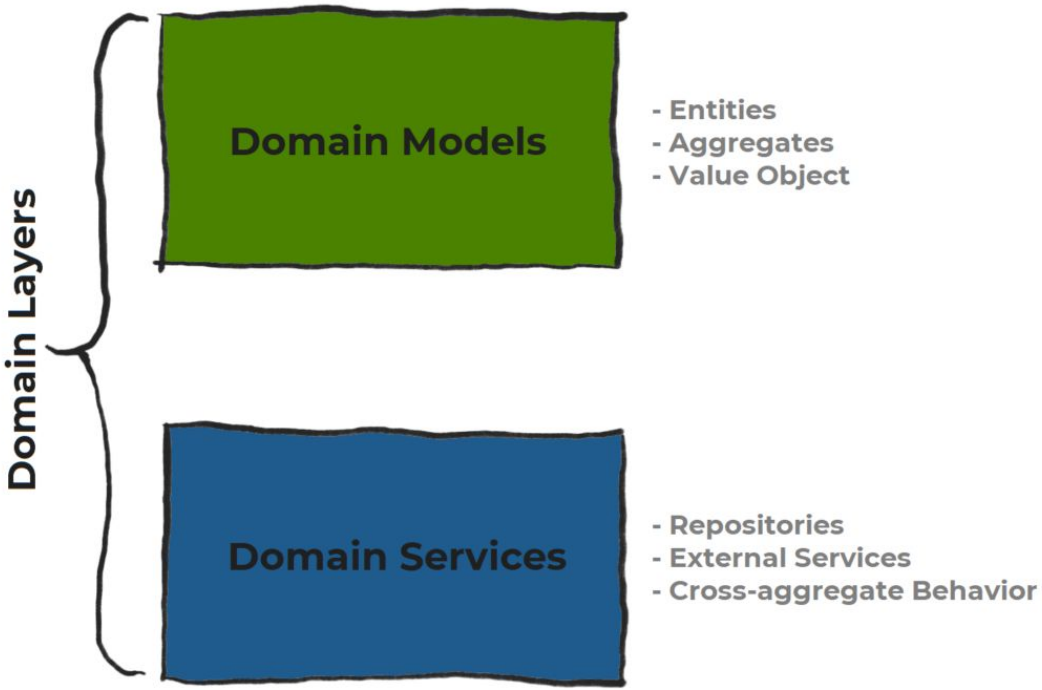


[Fonte: Edrawsoft.com - Software Simples de Diagrama de Arquitetura]



Fonte: Fowler, Martin (2014) - Bounded Context

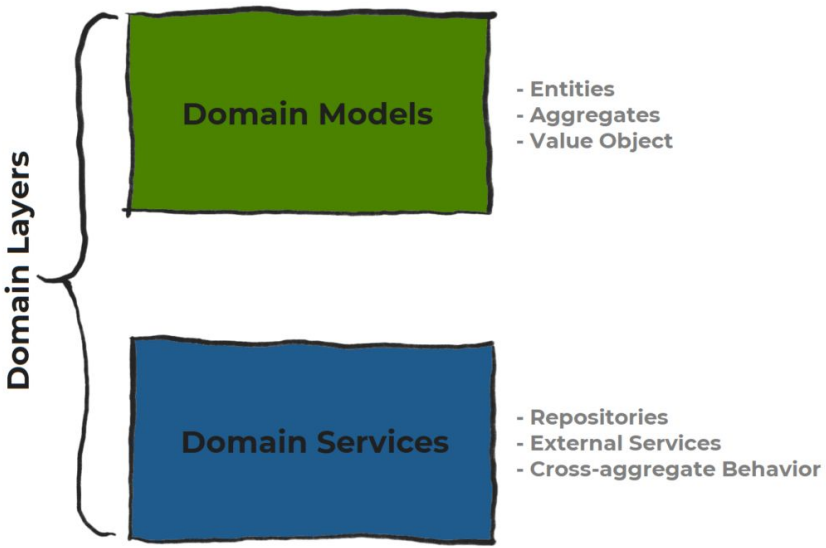
Design Tático



Conjunto de ferramentas a serem utilizadas na construção do modelo, aplicados a cada contexto delimitado.

Fonte: Zup(2022) - Domain-Driven Design: guia básico sobre DDD

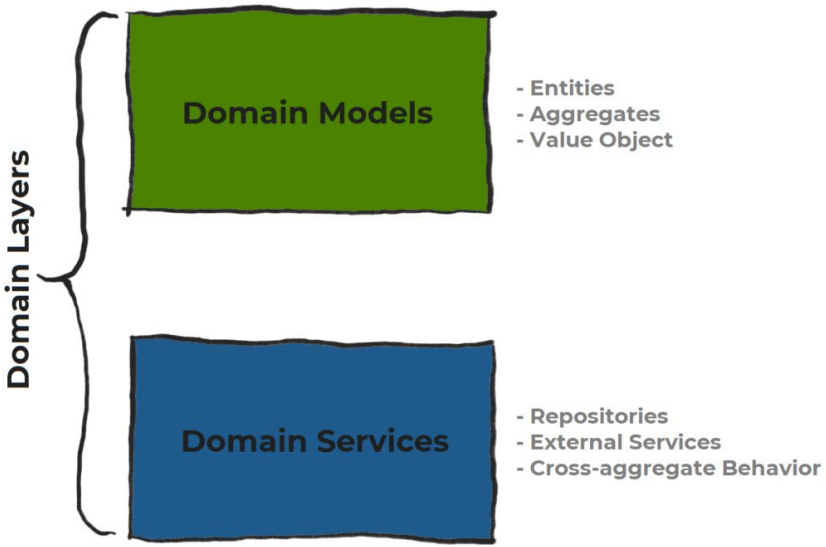
Design Tático



Domain Models: Conhecimento estruturado do problema, identificando os relacionamentos e funcionando como ferramenta de comunicação.

Fonte: Zup(2022) - Domain-Driven Design: guia básico sobre DDD

Design Tático



Domain Service: Estrutura sem estado que fornece comportamento do mundo dos negócios, atuando sobre as entidades e agregações, sendo relevante somente do ponto de vista do negócio.

Domain-Driven Design: Tutorial em .NET (devmedia.com.br)

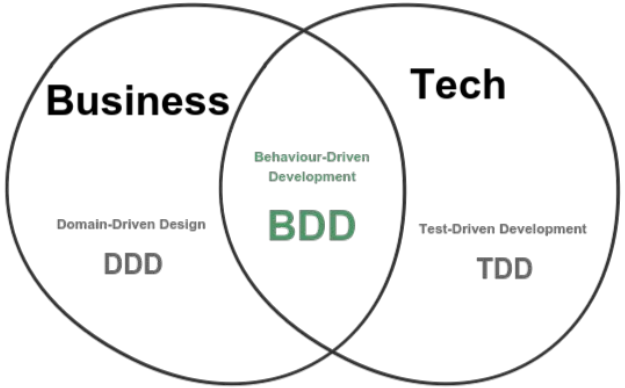
Fonte: Zup(2022) - Domain-Driven Design: guia básico sobre DDD

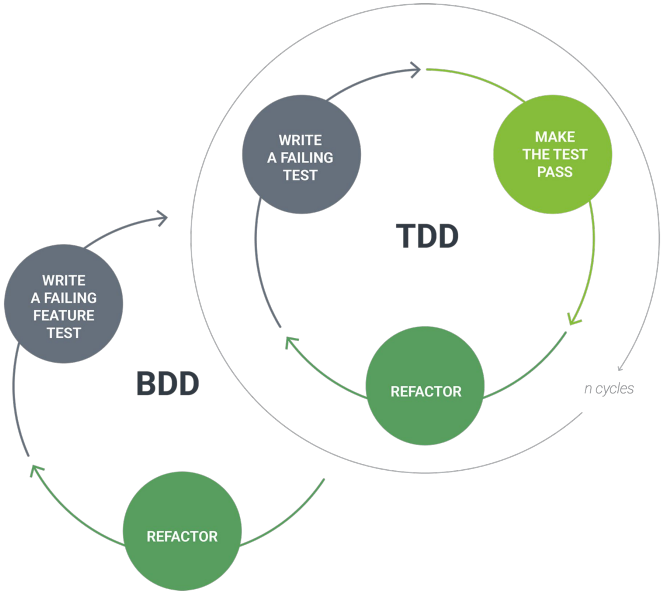
Aula 1 . Etapa 4

E como eles se integram?

// Entendendo as diferenças entre BDD, TDD e DDD

Como esses conceitos se conectam?





Fonte: Matola, Rodrigo - Quem escreve o BDD? (2020)

Entendendo as diferenças entre BDD, TDD e DDD

Carolina Santana Louzada

Analista QA - Venturus

Percurso

Aula 1

Afinal, qual a diferença entre BDD, TDD e DDD?

Dúvidas durante o curso?

> Fórum do curso
> LinkedIn: Carolina Santana Louzada
> Comunidade online (Discord)

SCAN ME