UNIVEM DIGITAL

Domain Driven Design (DDD)

Especialização em Desenvolvimento FullStack

Prof. Dr. João Ricardo Favan

07 de maio de 2022

Agenda

- Apresentação Professor
- Apresentação Disciplina
- Domain Driven Desing (DDD)
- Lean
- Agile
- Kanban
- Scrum

Apresentação do Professor

Graduado em Tecnologia em Informática para Gestão de Negócios

Especialista em T.I. em Saúde

Mestrado e Doutorado em Ciência Florestal (Inteligência Artificial)

Professor em disciplinas da área de Computação na Fatec Pompéia e na Univem.

Objetivos da Disciplina

Apresentar ao aluno os principais conceitos sobre o DDD (Domain Driven Design) e sua aplicação no desenvolvimento de projetos de software.

DDD?

Introdução

Domain Driven Design (Projeto Orientado a Domínio)

Eric Evans (2003)

Série de padrões de projetos

Domínio é o centro do projeto

Não é “Orientação a Objetos” mas compartilhá-se as boas práticas

Não falaremos de codificação ou linguagem.

Principios

Alinhamento do código com o negócio: Contato entre os desenvolvedores e especialista do domínio.

Favorecimento da Reutilização: os blocos de contrução facilitam o aproveitamento do mesmo conceito.

Mínimo de acoplamento: Interação entre as partes do sistema sem dependências entre modulos, classes, etc

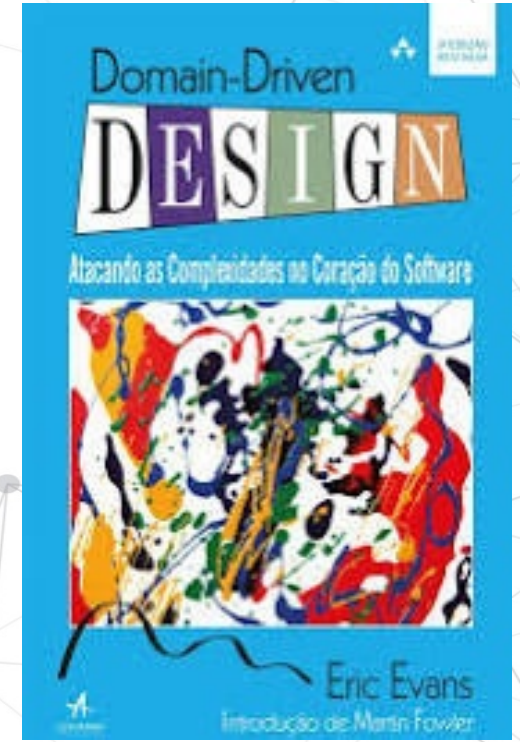
Independência da Tecnologia: O foco não esta na tecnologia, mas sim nas regras de negócio.

Principios, mas não só isso

Processos (Integração Contínua)

Relacionamento entre times

Sistemas complexos



Colocando o Modelo para Funcionar 3 Pilares

Linguagem Ubíqua

Contextos Delimitados

Mapa de Contextos

Colocando o Modelo para Funcionar

Linguagem Ubíqua

Linguagem comum, com termos bem definidos, que fazem parte do domínio do negócio e que são usados por todas as pessoas que fazem parte do processo de desenvolvimento de software

Glossário com termos que fazer parte do dia-a-dia dos especialista de negócios

Colocando o Modelo para Funcionar

Linguagem Ubíqua

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

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

Colocando o Modelo para Funcionar

Linguagem Ubíqua

Compreendida por todos e sem ambiguidades

Palavra única dentro do domínio

Readequações devem ser orais, escritas e codificadas

Colocando o Modelo para Funcionar

Linguagem Ubíqua

Utilizada para criação do modelo de domínio

Projeto Dirigido por Modelo (MDD)

Modelo abstrato deve ser a representação perfeita do domínio.

“Tudo que há no negócio deve estar no modelo. Tudo que esta no modelo deve haver no negócio.”

Colocando o Modelo para Funcionar

Criação do modelo abstrato deve ser feita em grupo, por todos os integrantes.

Processo contínuo de Maturação

The model guides the coding, and coding enhances the model

Blocos de construção do Model Driven Design (MDD)

Isolar o modelo de domínio das demais partes do sistema.

Arquitetura em Camadas

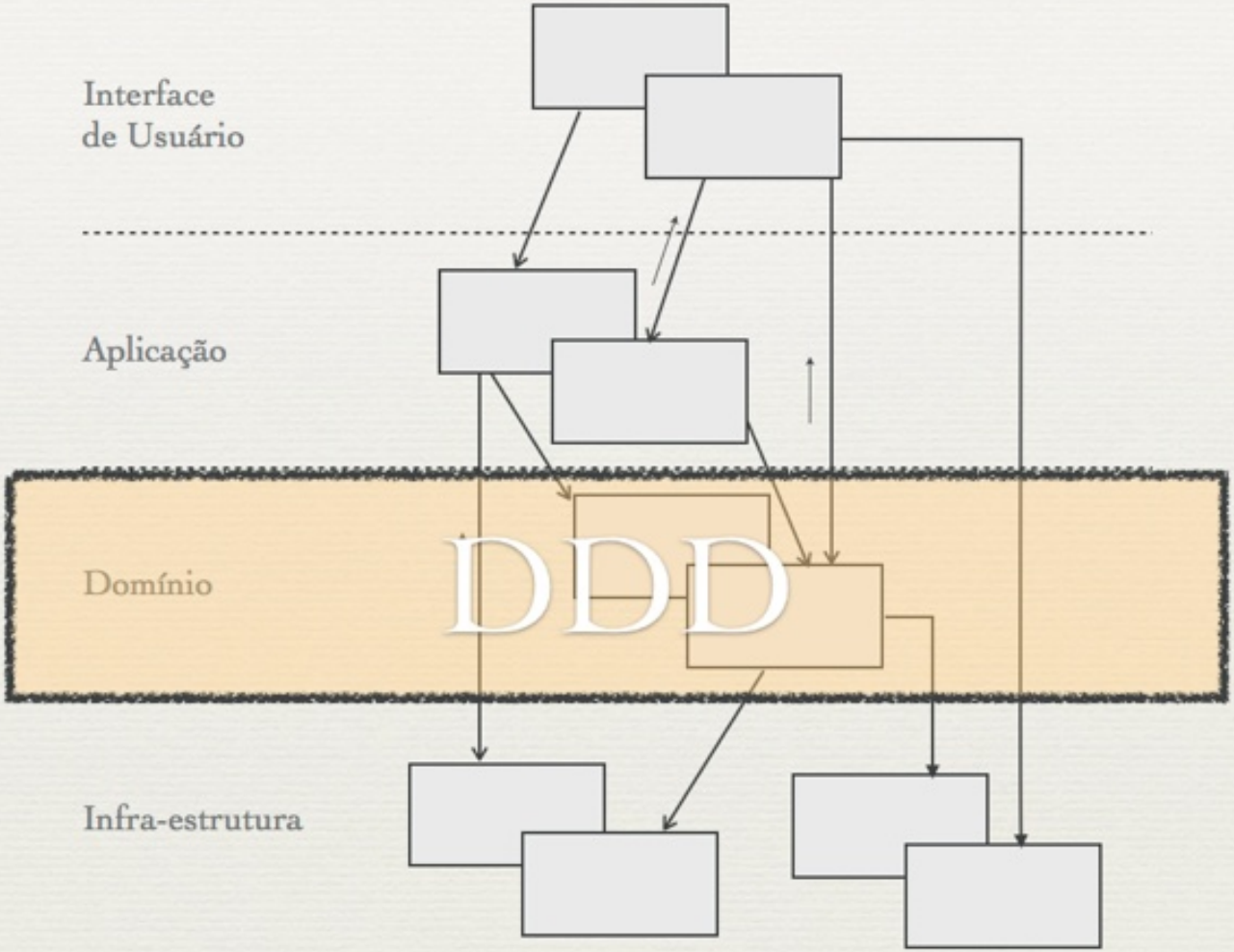
Blocos de construção do Model Driven Design (MDD)

Interface do Usuário: Exibição das informações

Aplicação: conecta IU com as camadas inferiores (não contém lógica de negócios)

Domínio: Representa os conceitos, lógicas e regras de negócio

Infra-estrutura: Recursos técnicos das camadas superiores (Conexões com BD, envio de mensagens)


Interface
de Usuário
Aplicação
Domínio
DDD
Infra-estrutura

Blocos de construção do Model Driven Design (MDD)

Entidades

Objetos de Valores

Agregados

Fábricas

Serviços

Repositórios

Módulos

Blocos de construção
Entidades

Classe de objetos que necessitam de uma identidade

Elementos do domínio que possuem ciclo de vida dentro da aplicação

Ex.: Cliente se cadastra na plataforma, faz login, faz as compras, se torna inativo, é excluído, etc

Blocos de construção
Objetos de Valores

Objetos que só carregam valores, mas que não possuem distinção de identidade (strings, números ou cores)

Objetos de Valores são imutáveis, uma vez criados, seus atributos internos não poderão mais ser modificados

Exemplo do lápis de cor. Exemplo do cartõa de crédito

Blocos de construção
Agregados

Compostos de Entidades ou Objetos de Valores que são encapsulados numa única classe.

O Agregado serve para manter a integridade do modelo.

Uma classe deve ser escolhida para servir de raiz do Agregado. Sendo sua manipulação somente através dessa classe.

Blocos de construção
Fábricas

Classes responsáveis pelo processo de criação dos Agregados ou dos Objetos de Valores

A criação dos agregados podem ser complexas, e, suas regras/lógicas não devem ser mantidas nas classes que a compõem

Extraímos as regras de criação para uma classe externa: a Fabrica.

Blocos de construção
Serviços

Classes que contém lógica de negócio, mas que não pertence a nenhuma Entidade ou Objetos de Valores.

Serviços não guardam estado

Blocos de construção
Repositórios

Classes responsáveis por administrar o ciclo de vida dos outros objetos, normalmente Entidades, Objetos de Valor e Agregados.

Os repositórios são classes que centralizam operações de criação, alteração e remoção de objetos

Agnósticos ao Banco de Dados

Blocos de construção
Módulos

Abstração para agrupar classes por um determinado conceito do domínio.

Não deve-ser agrupar elementos segundo conceitos de infra-estrutura.

Exemplo do paciente

Refatorando para compreender a profundamente o modelo

Após elaboração do modelo que reflita seu domínio, utilizando os blocos de construção MDD, o modelo deve ser aperfeiçoado continuamente.

O modelo deve ser refatorando sempre que houver maior compreensão dos conceitos importantes do domínio.

Alguns padrões podem ajudar.

Interface de Intenção Revelada

Usar o “o que” e não o “como” para os nomes das classes e métodos.

Funções sem efeitos-colaterais

Priorizar Métodos que não alteram o estado dos objetos, priorizando estas alterações para os comandos.

Asserções (assert)

Nos comandos que alteram o estado dos objetos, criar testes de unidade que rodem automaticamente, ou introduzir “asserts” no código que validem as alterações de estado esperada.

Projeto Estratégico

Aumentá-se a complexidade do projeto

Aumenta-se os contextos, as interações entre sistemas

Como verificar o que tem maior valor para o negócio?

Como se comunicar com os outros times de desenvolvimento?

Projeto Estratégico

Padrões para segmentar o software em várias partes, denominada **contexto**.

Cada **Contexto Delimitado** deve estar claro para todos os envolvidos.

A **fronteira** entre os contextos deve ser clara para todos

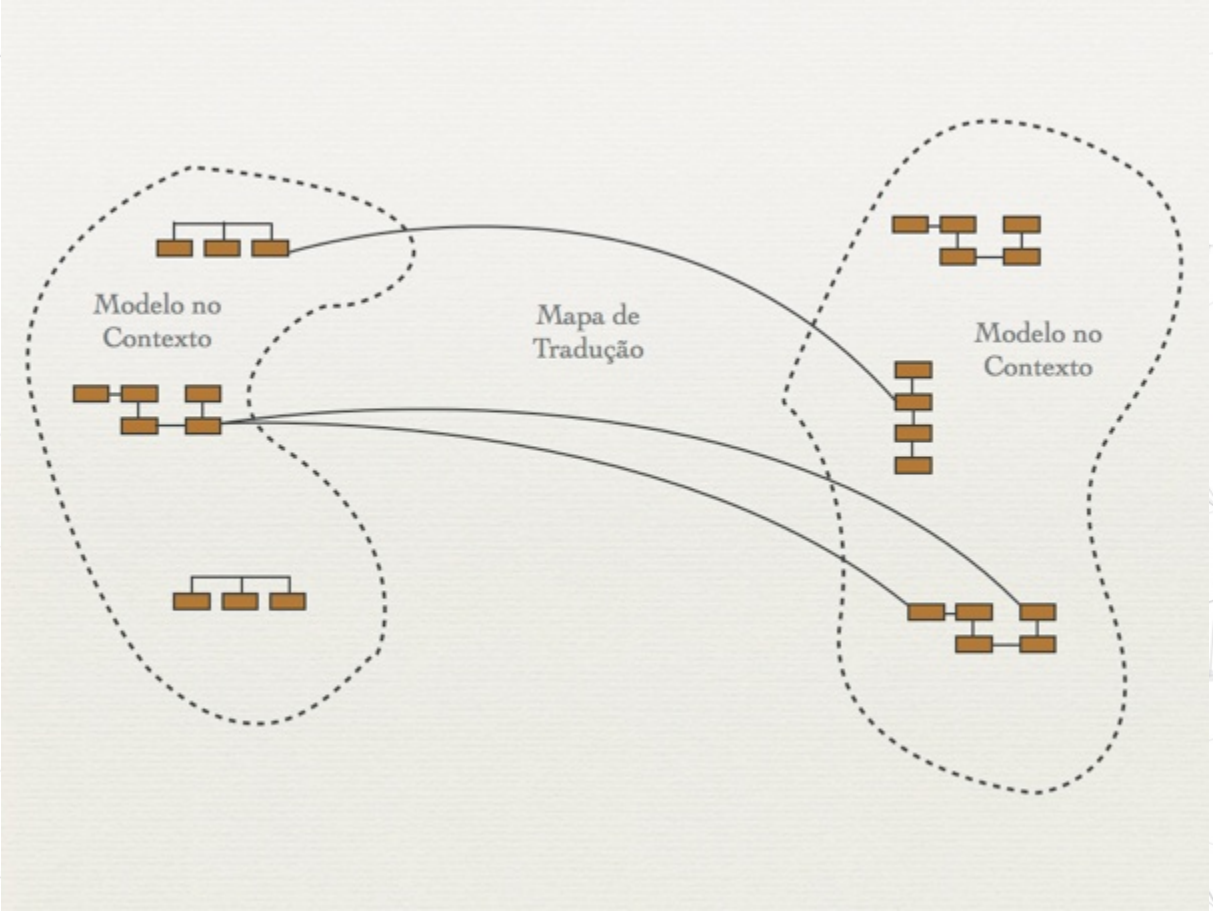
Mapa de Contexto

Ressalta-se os componentes do software e interação da equipe

Mapa de tradução que facilitará a comunicação entre as equipes de desenvolvimento

Contém os termos utilizados por cada equipe e seu equivalente para a outra equipe

Mapa de Contexto



Mapa de Contexto

Ressalta-se os componentes do software e interação da equipe

Mapa de tradução que facilitará a comunicação entre as equipes de desenvolvimento

Contém os termos utilizados por cada equipe e seu equivalente para a outra equipe

Mapa de Contexto (Relações)

Produtor-Consumidor: Produtor fornece software para o consumidor. O produtor assume um compromisso que o consimidor considera que irá cumprir.

Produtor deverá sempre fazer testes automatizados para garantir a integridade da interface

Mapa de Contexto (Relações)

Conformista: Caso não seja possivel alterar parte do sistema mantido por outro time.

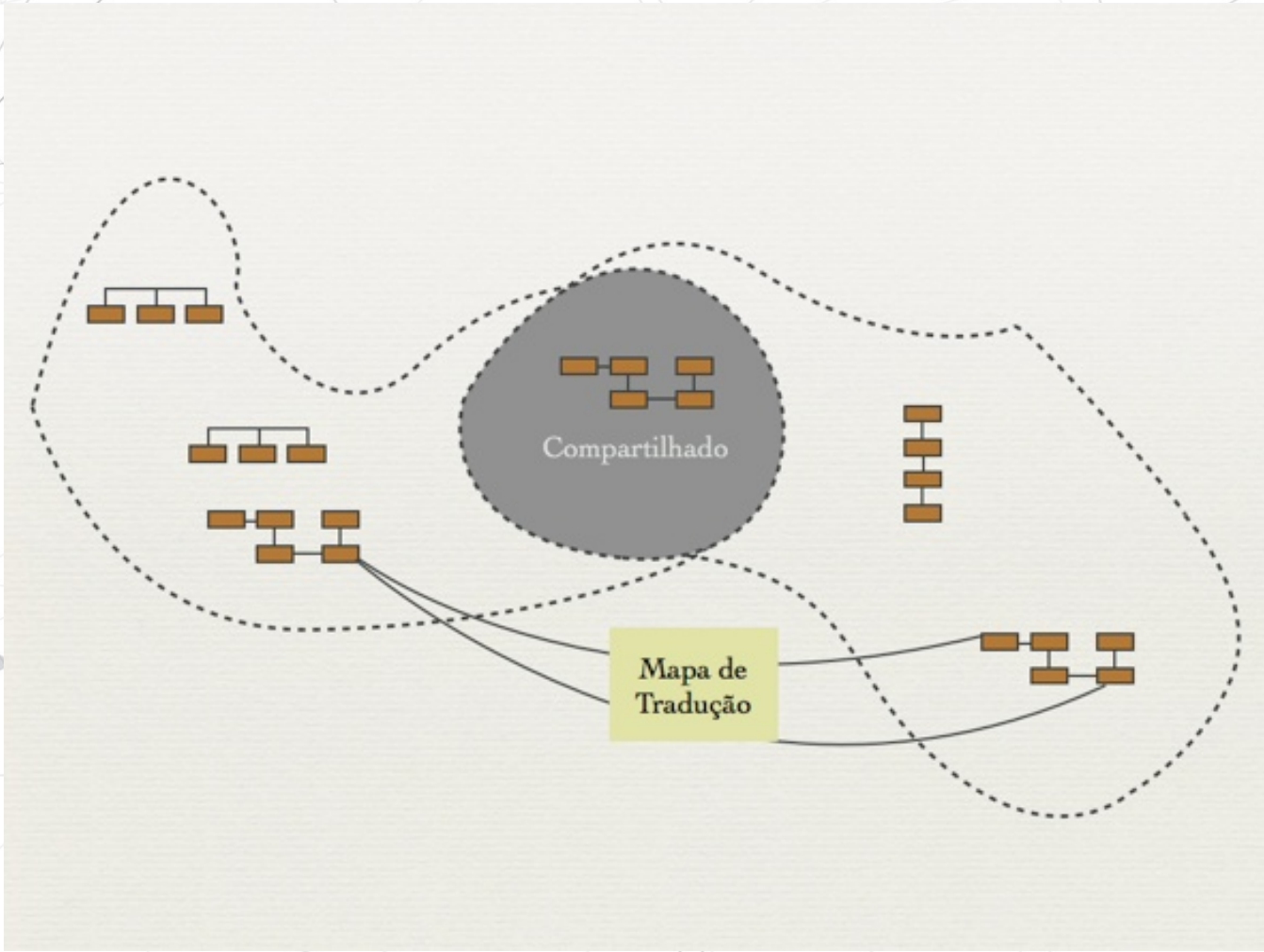
Não se solicita alterações na funcionalidade do outro sistema. Alteramos o “nosso” sistema para se adequar ao que foi oferecido.

Mapa de Contexto (Relações)

Núcleo Compartilhado: Dois “times” alteram uma mesma base de código comum.

Essa base deve estar bem definida e possuir muitos testes automatizados

Alterações devem ser comunicadas e os testes devem fazer parte do processo de **integração contínua**

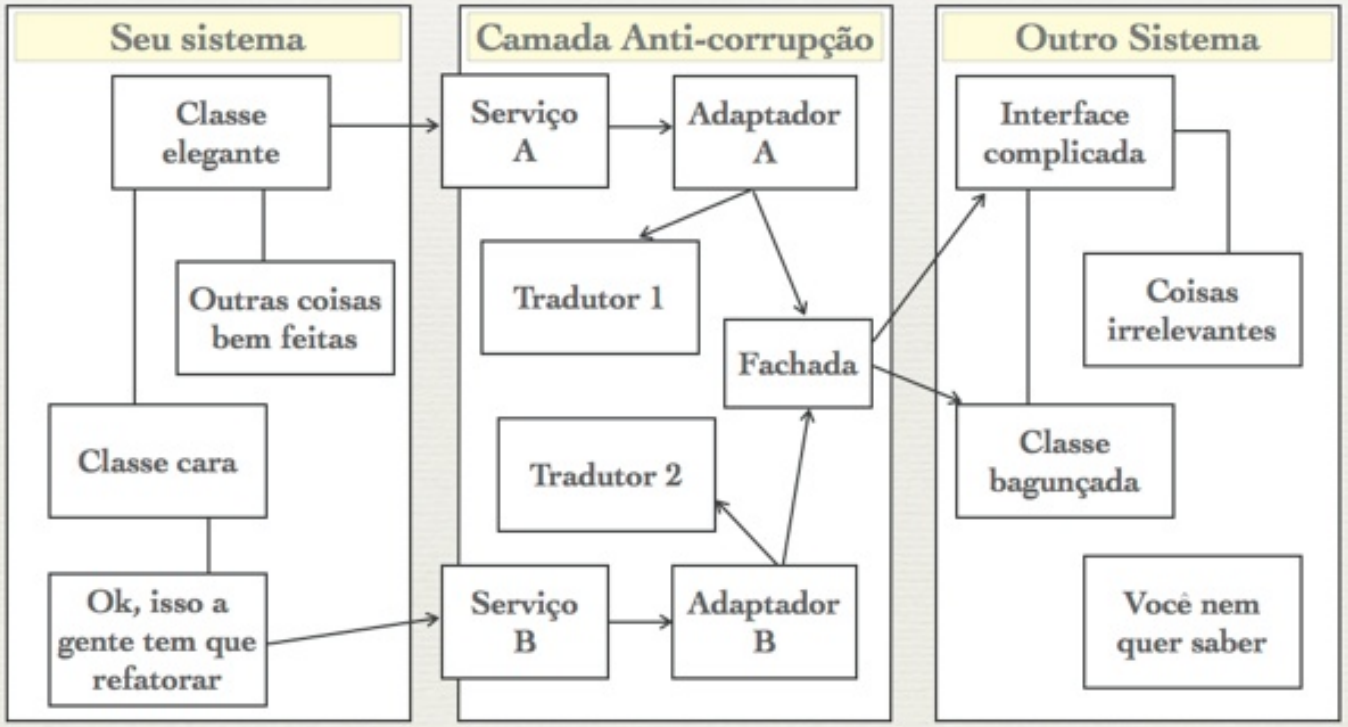

Compartilhado
Mapa de
Tradução

Mapa de Contexto (Relações)

Camada Anti-Corrupção: Camada intermediária para tradução e adaptação das chamdas para outros sistemas.

Pode ser utilizados em interfaces com sistemas legados ou para padronização em sistemas externos.

Camada Anti-corrupção



Mapa de Contexto (Relações)

Caminhos Separados: Desenvolvimento é feito de tal forma que os sistemas (contextos) não dependam um do outro.

Utilizado em partes periféricas do domínio

Mapa de Contexto (Relações)

Serviço Aberto de Funcionalidades: Vários clientes interagem com uma mesma parte do sistema.

Padronização na forma de interação com os clientes.

Elaboração e Documentação de uma API que será utilizada pelos clientes

Ex.: Google, Flickr, Twitter, Amazon

Em Busca do Núcleo

Destilação do Domínio: processo que vai refinando o domínio até que não se tenha mais nada a retirar, chegamos ao **Núcleo do Domínio** (core domain).

O processo consistem em consecutivas separações de módulos, refatorações, extração de métodos, classe, conceitos.

Os Conceitos centrais do nosso negócio (core bussness) são o núcleo do domínio.

Em Busca do Núcleo

Domínio Genérico: tudo aquilo que não faz parte do núcleo e deve ser organizado segundo seu contexto. Podemos ter vários domínios genéricos em uma aplicação

O núcleo do domínio sempre tem prioridade sobre os domínios genéricos

Em Busca do Núcleo

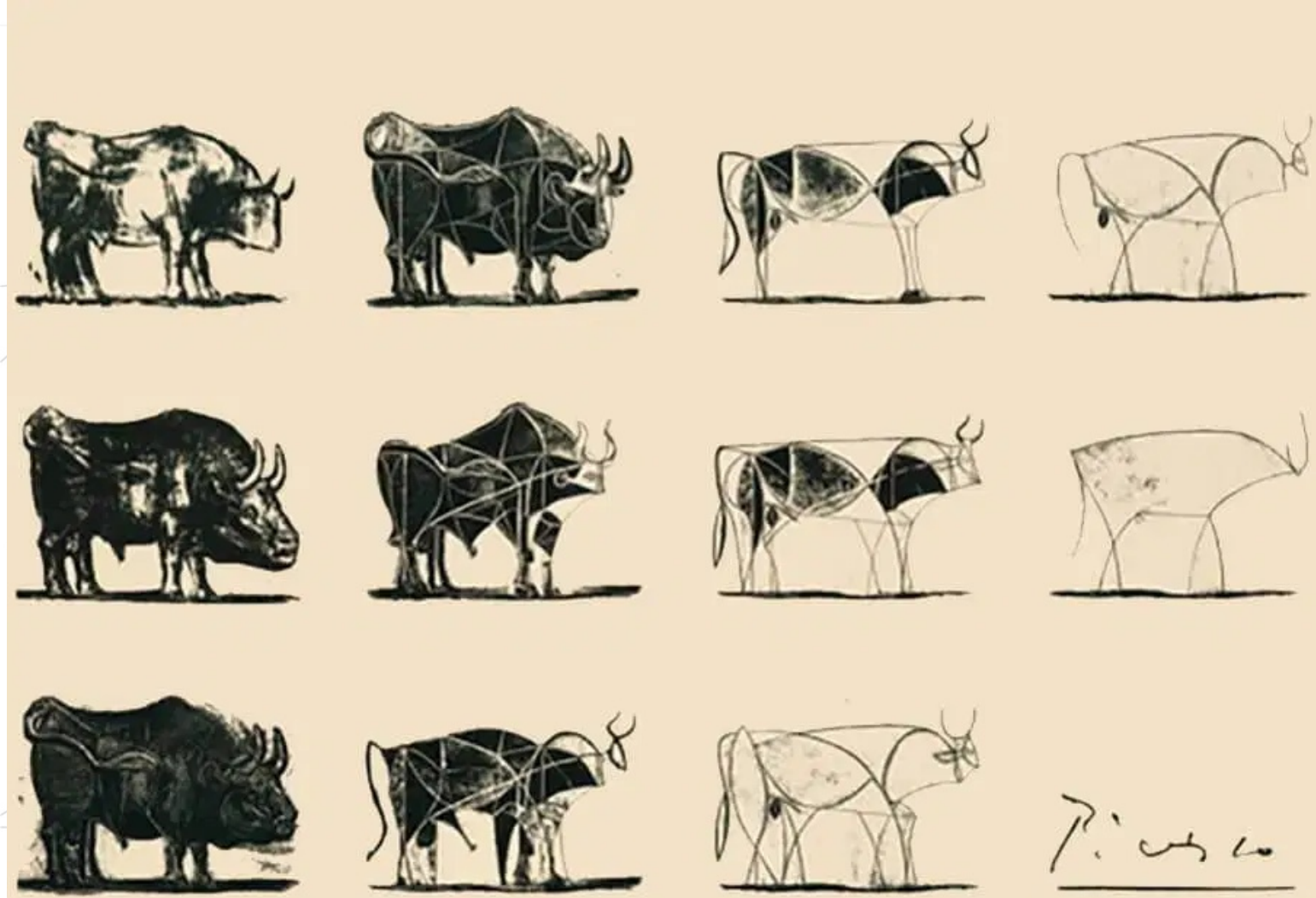
Sentença da Visão do Domínio:

“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.”

Em Busca do Núcleo

Não faz parte do núcleo 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.”



Ok, mas vamos a um exemplo...

Ok, mas vamos a um exemplo...

Linguagem Ubíqua – Exemplo de Glossário

Playback: Processo de execução de um vídeo

Vídeo: Conteúdo que será visto pelo usuário

Capa: Figura descritiva de um vídeo

Perfil: Perfil gerado pelo usuário para separar os utilizadores e suas preferências e “Minha Lista”

Minha Lista: Lista de vídeos usada pelo usuário para armazenar os vídeos que tem interesse em ver depois

Título: Nome do vídeo

Pessoas: Personagens que fazem participação em determinado vídeo

Gênero: Estilo que define um tipo de vídeo. Ex.: drama, terror, comédia

Catálogo: Catálogo de vídeos disponibilizados

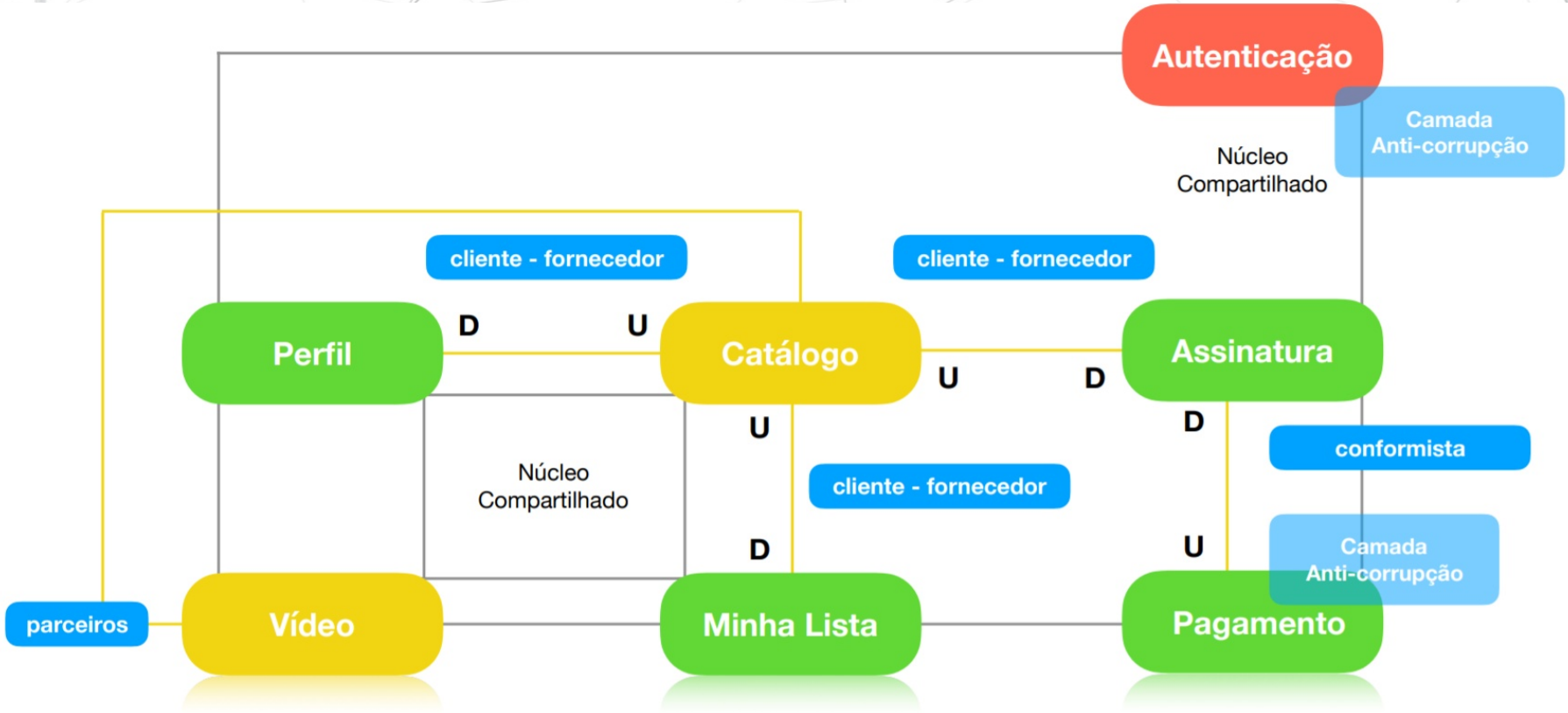
Ok, mas vamos a um exemplo...

Bounded Contexts – Exemplo de História

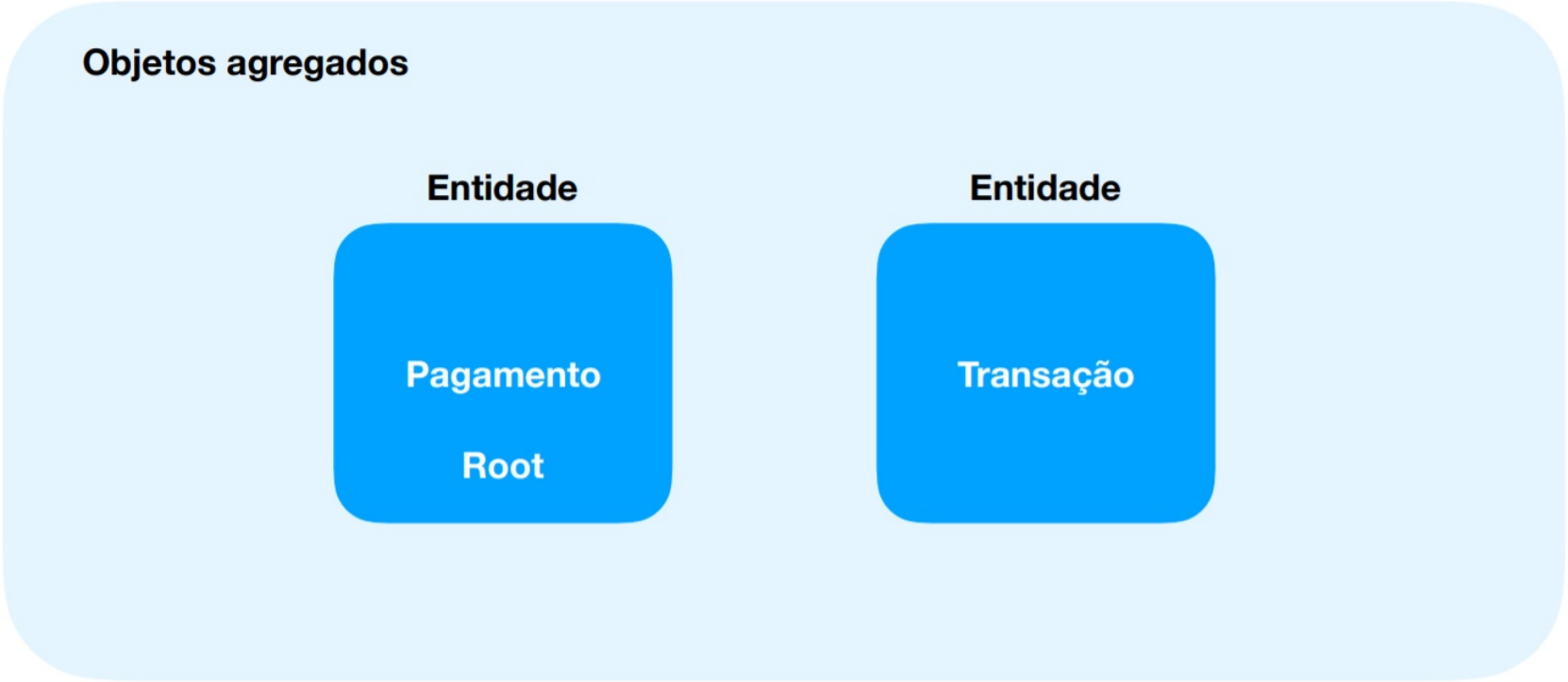
1) Cria uma conta para acessar o sistema
2) Faz o login
3) Seleciona o plano de acesso e realiza a assinatura
4) Cria um perfil de acesso
5) Navega no catálogo de vídeos
6) Realiza uma busca através do título e/ou gênero
7) Adiciona os vídeos, que tem interesse, em sua lista
8) Acessa o vídeo escolhido e faz o playback

Modelagem Estratégica

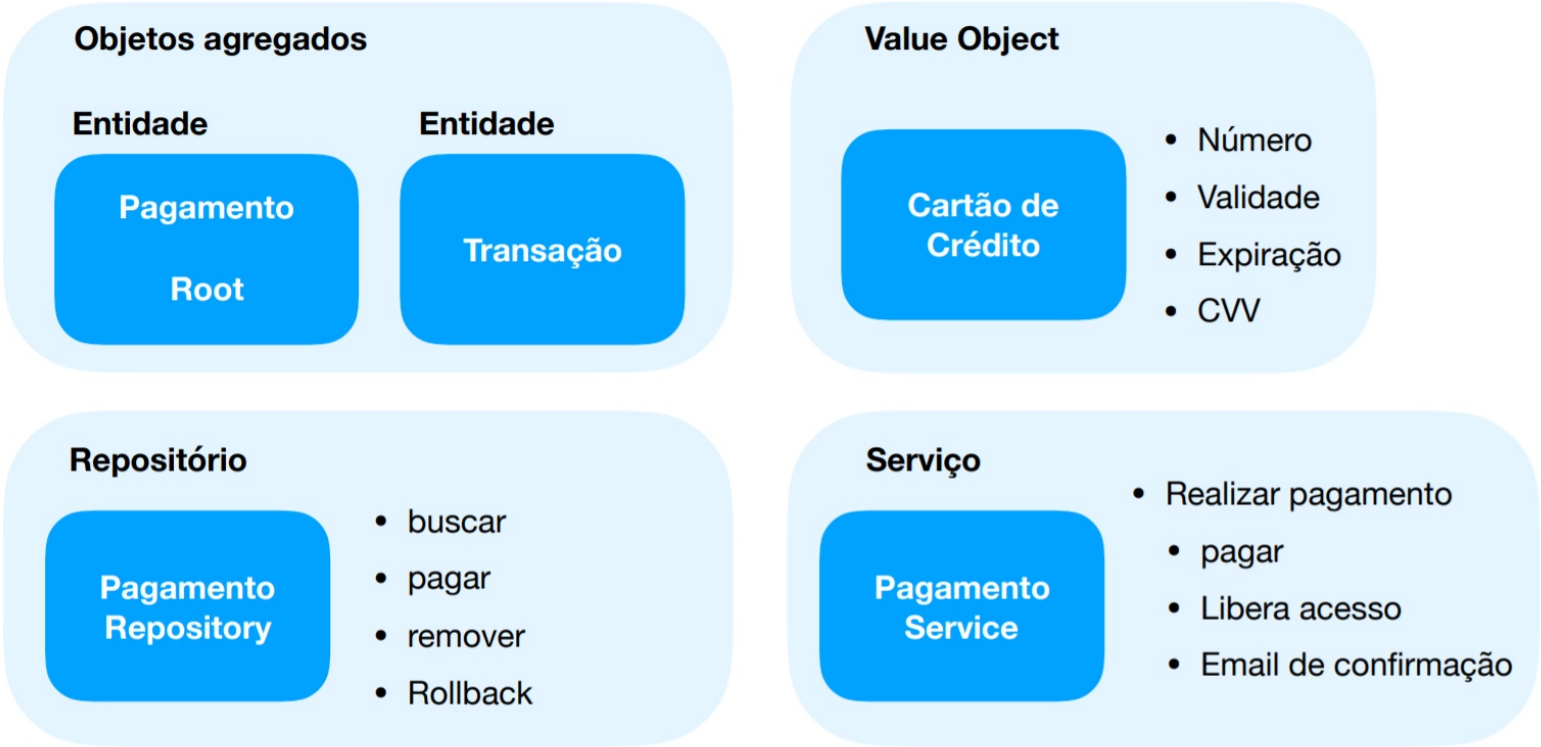
Mapa de Contexto



Domain Model Patterns



Consolidado



UNIVEM DIGITAL

Referências

AGILEMANIFEST. **Manifesto para Desenvolvimento Ágil de Software**. Disponível em: <http://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em: 21 maio. 2021.

CUKIER, D. **DDD – Introdução a Domain Driven Design** | AgileAndArtAgileAndArt, 16 jul. 2010. Disponível em: <http://www.agileandart.com/2010/07/16/ddd-introducao-a-domain-driven-design/>. Acesso em: 21 maio. 2021

EDUARDO. **Scrum** - Primeiros Passos: Cerimônias e PapéisSciensa, 11 ago. 2017. Disponível em: <http://blog.sciensa.com/scrum-primeiros-passos-cerimonias-papeis/>. Acesso em: 21 maio. 2021

ENGELBRETH, E. X. **Agile x SCRUM x Lean**, o que cada um é?FIXE, abr. 2017. Disponível em: <https://youwilldobetter.com/2017/04/agile-x-scrum-x-lean-o-que-cada-um-e/>. Acesso em: 21 maio. 2021

ESPINHA, R. G. **O que é Kanban:** Guia completo (atualizado 2021). Disponível em: <https://artia.com/kanban/>. Acesso em: 21 maio. 2021.

EVANS, E. **Domain-Driven Design**: atacando as complexidades no coração do software. 3. ed. São Paulo: Alta Books, 2016.

LASERMEC. **Lean Agile:** 5 melhores práticas para implementar essa metodologia. Disponível em: <https://blog.lasermec.com.br/lean-agile/>. Acesso em: 21 maio. 2021.

SUTHERLAND, J. **Scrum**. A Arte de Faze o Dobro do Trabalho na Metade do Tempo. Tradução: Natalie Gerhardt. 1. ed. São Paulo: LeYa, 2014.

WILLIANS, W. **O que é DDD** - Domain Driven DesignFull Cycle, 15 ago. 2019. Disponível em: <https://fullcycle.com.br/domain-driven-design/>. Acesso em: 21 maio. 2021