VLADMIR CRUZ

PDS TECH

SOFTWARE ARCHITECTURE

DOMAIN-DRIVE DESIGN

AULA 03

SUMÁRIO

O QUE VEM POR AÍ?

Nesta aula, veremos como identificar o desafio do negócio, como entender o problema, como criar uma linguagem única para que possamos nos comunicar com o negócio, como modelarmos um domínio e seus subdomínios e como criar as ferramentas que vamos utilizar para nos comunicar e documentar com o cliente. Além disso, falaremos dos Contextos Delimitados e sua importância no DDD.

HANDS ON

Na aula 03, falamos sobre o domínio do negócio. Para isso, utilizaremos a linguagem ubíqua, contextos delimitados e, claro, quais ferramentas nos ajudam neste cenário. Assista a videoaula que preparamos com muita dedicação para você aprender ainda mais sobre esses assuntos.

Voltando ao nosso exemplo de escola, vamos definir o desafio do negócio, em que trabalhamos em uma área da escola, afinal, usar toda a escola deixaria o exemplo gigantesco. Vamos trabalhar com a questão das entregas de atividades e suas correções.

Quais os desafios nesta área? Como criarmos a linguagem ubíqua desse processo? E como o ator “pais” é representado de formas completamente diversas entre várias áreas?

SAIBA MAIS

Descoberta e formação do conhecimento

Já temos um desafio (construir os sistemas para uma escola), já falamos de Domínios, Subdomínios, Domain Experts e uma das técnicas para mapear histórias. Mas como vamos entender e combinar isso tudo?

Nossa escola tem várias áreas de “negócios”, cada uma com suas particularidades e dialetos próprios. E, como falamos anteriormente, DDD é sobre entender o que é a demanda do negócio, aprender o que agrega valor.

O desafio do negócio

Quando falamos sobre o desafio do negócio, estamos endereçando o que o negócio faz, quais são os seus processos e quais os desafios conhecidos.

O ponto não é somente elencarmos o óbvio, há muito mais que isso, pois em momento algum temos a percepção real do nosso Domain Expert, só o desenho do que acontece hoje (lembram do desejo vs realidade?).

Então, vamos nos aprofundar um pouco mais, pois em uma conversa com o Domain Expert temos uma visão dos problemas enfrentados pelo negócio.

Porém, somente seremos capazes de desenhar esse modelo com uma comunicação clara, e isso é uma via de mão dupla: o Domain Expert tem que falar algo que os desenvolvedores possam entender e vice e versa. Mas como fazemos isso? É o que vamos falar agora!

A linguagem Ubíqua

Durante a descrição do cenário, o Domain Expert começa a usar termos como próprios do negócio, como testes, notas, sistema, e muitos outros que achamos que entendemos. Por exemplo, os pais de uma criança que ainda estejam em processo de escolher uma escola são vistos de forma diferente pelos departamentos da mesma escola, tais como: admissão, marketing e secretaria.

Por isso, é de extrema importância que tenhamos os termos claros para cada cenário e subdomínio. Dessa forma, tanto o negócio quanto o time de desenvolvimento “falam a mesma língua” e, posteriormente, esses termos podem e serão utilizados no desenho do sistema.

Esses elementos fazem parte do que chamamos de linguagem ubíqua, que é a linguagem que utiliza as terminologias da realidade do negócio.

É importante notar que a linguagem ubíqua deve ser consistente e detalhada, facilitando assim a comunicação e o entendimento, e vale ressaltar dois pontos que podem ocorrer no processo de mapeamento:

Termos Ambíguos: dentro de um subdomínio, podemos ter um mesmo termo com vários significados. Por exemplo: política. Esse termo pode significar uma lei regulatória ou uma regra interna da escola. Como em linguagem ubíqua precisamos de uma definição para cada termo, aqui podemos definir que o termo política será utilizado para os dois significados.

Termos Sinônimos: dentro de um subdomínio, podemos ter um termo que é utilizado para vários significados que possuem muito mais detalhes. Por exemplo, em TI utilizamos sempre o termo login para referenciar o ato de efetuar autenticação no sistema, ou para referenciar a conta do usuário. É importante notar que são coisas totalmente diferentes que alocamos o mesmo nome por conveniência. Em linguagem ubíqua, sempre que possível, temos que quebrar esses termos e lhes dar definições únicas e específicas para evitar problemas futuros.

Modelagem do Domínio

Ao criar um modelo do Domínio, estamos desenhando uma abstração de um processo do qual queremos resolver um problema.

Essa documentação é essencial para que possamos conduzir as conversas com os Domain Experts e extrair tudo que é necessário para entender o problema. A linguagem ubíqua que vimos anteriormente é um passo muito importante no alinhamento da comunicação entre as partes.

É importante ressaltar que, na medida em que avançamos no desenho do modelo, também teremos refinamentos. O que discutimos anteriormente ao falar da

Realidade vs Desejo, será aplicado ao iterarmos sobre o modelo, como o foco no que é importante, termos uma linguagem única (ubíqua) vai nos ajudar a criar um modelo mais efetivo e a criar uma solução que reflete o que foi dito.

Existem diversas ferramentas que podem ser utilizadas para catalogar os modelos e para manter um dicionário de linguagem ubíqua. Podemos citar aqui as Wikis, ferramentas de colaboração como o Notion.io e outras mais. Independente da ferramenta, é importante manter tudo atualizado na medida em que as interações acontecem.

Ferramentas

Como vimos até aqui, é muito importante que tenhamos controle de tudo que estamos fazendo, ou seja, que cataloguemos tudo, de forma que posteriormente possamos compartilhar e manter tudo atualizado na medida em que seguimos em frente.

Não há uma “fórmula mágica” de como fazer isso, mas precisamos montar uma estrutura que nos dê suporte e seja fácil de se trabalhar, e isso significa pensar em todos os envolvidos no processo de desenvolvimento.

Um bom exemplo é montarmos uma Wiki que consolida os diversos recursos, tais como os listados abaixo:

- Wiki para ser a central do projeto, consolidando todos os recursos.
- Seção na Wiki com o a descrição do projeto.
- Seção na Wiki com o time do projeto (se tivermos mais de um time). Devemos então criar diversas subpáginas para contemplar cada parte do projeto. Em outras palavras, uma página para cada subdomínio).
- Seção na Wiki para a linguagem ubíqua (esse é o nosso “Dicionário”).
- Seção na Wiki para os cenários que criamos, suas premissas e limitações.
- Link para um repositório do GitHub, com os códigos do projeto e sua documentação.
- Link para a Ferramenta de Gestão do Projeto.

É uma estrutura simples, mas que fará toda a diferença no andamento do projeto.

Contextos Delimitados

Até aqui falamos de entender o negócio trabalhando em conjunto com o Domain Expert para que possamos desenhar a solução a ser implementada, que é baseada em processos de negócio.

Identificamos o domínio e seus subdomínios, como cada um “participa” do negócio e qual a sua parte na estratégia da empresa.

Pois bem, agora começamos a criar limites para a nossa solução, limites que não são definidos pelos subdomínios. Esses limites existem dentro de contextos que identificamos ao analisar o negócio, são o que chamamos de Contextos Delimitados (Bounded Contexts).

Temos alguns pontos importantes:

- Não há uma regra para definir o tamanho de um contexto delimitado, tudo depende da análise do arquiteto. Como falamos anteriormente, a linguagem ubíqua é uma ótima métrica, se a terminologia é a mesma em dois contextos e tratam os processos de negócio de formas muito parecidas, pode ser uma boa ideia juntar as duas no mesmo contexto, ou até mesmo o contrário; se forem muito distintos, separe-os para que possam ser implementados sem interferência.
- A quantidade de recursos não é fixa, depende do tipo de projeto que você tem. O tamanho de um contexto pode dizer muito sobre isso, mas não é porque um contexto é grande demais que você precisa de um time imenso. Muitas coisas podem ser dependentes de outras e não adianta abrir um paralelo no desenvolvimento, pois não acelerará nada.
- O número de contextos pode mostrar o número de times que farão parte da equipe de desenvolvimento, mas como vimos acima, podemos refinar sucessivamente nosso modelo e ir integrando cada vez mais nossos contextos, diminuindo assim a quantidade.

- Podem existir casos de contextos delimitados que englobem a solução inteira, se a solução for muito pequena isso é possível, assim como podemos ter contextos delimitados em que não haverá agrupamento de contextos e será 1 para 1, no caso de sistemas muito grandes e complexos.

Importante ressaltar: um contexto delimitado é sempre trabalhado por um time, nunca compartilhado entre dois, mas um time pode trabalhar em diversos contextos.

A figura 1 – “Times trabalhando no contexto delimitados” deixa isso evidente, usando o nosso caso:

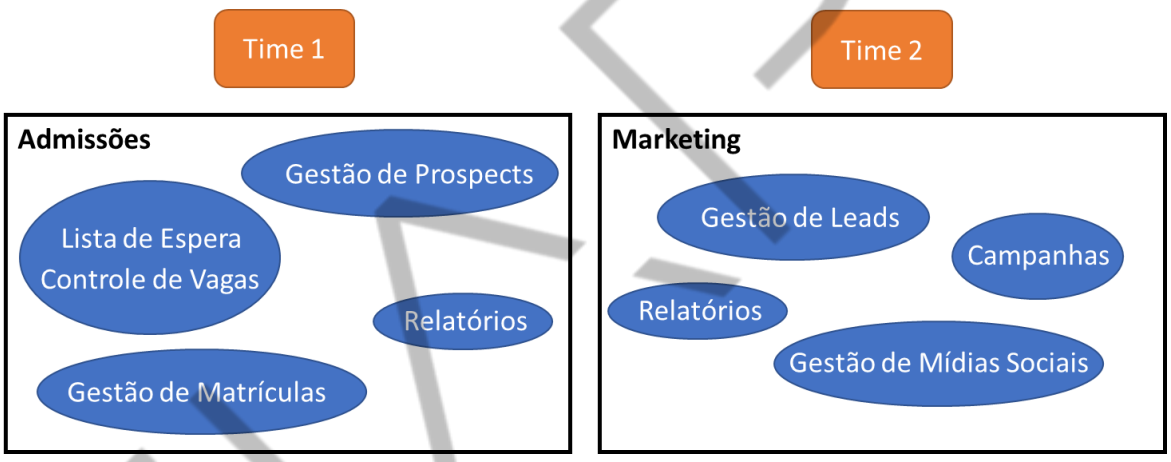

Figura 1 – Times trabalhando nos Contextos Delimitados
Fonte: Elaborado pelo autor (2023)

Isso ocorre, pois, os times criam uma linguagem própria para se comunicar, que muitas vezes está atrelada à linguagem ubíqua e aos processos de negócio que estão trabalhando.

Contextos delimitados são limites que impomos em um modelo de domínio. Esses limites contêm termos e definições, propriedades e operações que compartilham uma linguagem única, a linguagem ubíqua.

O QUE VOCÊ VIU NESTA AULA?

Nesta aula, falamos de como identificar o desafio e mapear o nosso domínio, criando uma linguagem única: a linguagem ubíqua. Ela garante que todos os elementos que sejam criados tenham um significado comum a todos que estão envolvidos, garantindo assim que não tenhamos problemas de significado ou entendimento.

Falamos também das ferramentas para mantermos nosso dicionário de linguagem ubíqua, e outros documentos que utilizaremos, e por fim, mas não menos importante, entendemos o que são contextos delimitados, sua função, e como podemos criá-los usando a linguagem ubíqua.

Não se esqueça que estamos no Discord para acompanhar a sua jornada aqui na Pós-Tech! Participe para tirar dúvidas, fazer networking e muito mais.

REFERÊNCIAS

COCKBURN, A. **Writing Effective Use Cases**. [s.l.]. Addison Wesley, 2001.

EVANS, E. **Domain-Driven Design, Tackling Complexity in the heart of Software**. Boston:. Pearson Education, 2003.

FOWLER, M. **Patterns of Enterprise Application Architecture**. [s.l.]. Addison-Wesley, 2002.

KHONONOV, V. **Learning Domain-Driven Design**. Sabastopol: O'Reilly Media. 2021.

MILLET, S. Tune, Nock. Patterns, **Principles and Practicves of Domain-Driven Design**. Indianapolis: Wrox, 2015.

VERNON, V. **Implementing Domain-Driven Design**. Boston: Pearson Education, 2013.

PALAVRAS-CHAVES

DDD. Modelagem de negócio. Linguagem ubíqua.