Jornada AppSec

Nivelando os conceitos de subdomínio e contexto delimitado

Autor: Felipe Pinheiro

GitHub: https://github.com/pinheiro-felipe/jornada-appsec-criando-api-de-postagem-de-status-do-pedido-para-delivery

10- Nivelamento de conceitos de subdomínio e contexto delimitado: Quais são os conceitos de subdomínios, contextos delimitados e mapeamento de contexto que precisamos aprender?

Contexto delimitado: Seria a divisão de um domínio em partes menores que possuem suas próprias regras, modelos de negócio e interagem entre si para no todo formar o domínio. Cada parte dessas seria um contexto delimitado, ou seja, uma área com uma fronteira que define suas limitações, linguagem ubíqua, as intenções de suas entidades e o que cabe somente a ela e a nenhum outro contexto delimitado.

De grosso modo podemos pensar no planeta terra e sua divisão em países. Cada país possui suas leis, cultura, linguagem e etc. Falamos praticamente as mesmas coisas, comentamos os mesmos assuntos de maneiras diferentes e ainda nos comunicamos uns com os outros a nível de países. Para essa comunicação acontecer existem algumas opções:

Usar uma linguagem comum, no caso o inglês.
Usar a linguagem de um dos países
Cada um falar a sua própria língua e usa um tradutor.

A comunicação possui seus desafios.

Voltando ao contexto delimitado, podemos entender que para a comunicação acontecer um contexto deve informar ao outro que algo importante para o negócio aconteceu. Cada contexto possui um evento limite que é relevante para o negócio e responsável por fazer a transição entre as diferentes fases.

Para chegarmos aos nossos contextos delimitados, devemos decompor o nosso domínio em subdomínios primeiro, que geralmente refletem a estrutura de como o negócio está organizado.

Domínio: Representa o que a empresa faz, a sua atividade. O domínio possui o domain expert (especialista de domínio) que é a pessoa que entende do negócio e apoia os times de desenvolvimento na modelagem do próprio domínio, definição das regras de negócio e linguagem ubíqua.

Subdomínio: É a divisão do domínio de acordo com as atividades ou fases dentro do negócio. A divisão do domínio em subdomínios pode ser mais complicada do que parece e uma dica importante é prestar atenção em quando muda o domain

expert (especialista de domínio), que é a pessoa que entende as regras e a comunicação usada.

Alguns subdomínios são mais importantes para o negócio que outros:

Subdomínio Principal (Core subdomain): Representa o conjunto de atividades chaves do domínio que se está tentando representar. Traz uma vantagem competitiva para a empresa, além de a diferenciar no mercado. Exige maior investimento, aprimoramento e comprometimento dos recursos. Seu aprimoramento traz complexidade e mudança para o negócio ser competitivo. Pode quem sabe, até ser extraído da aplicação e ser transformado num novo negócio.

Subdomínio de suporte (Supporting subdomain): Serve para dar suporte a operação principal, ao subdomínio principal que não pode ser bem-sucedido sem ele.
Não possui o mesmo nível de investimento do subdomínio principal e por isso pode-se internalizar seu desenvolvimento ou considerar terceirizar, evitando assim onerar recursos que poderiam ser empregados no subdomínio principal.

Subdomínio genérico (Generic subdomain): Executa operações que são importantes para o domínio, mas não tem relação direta para sustentar a operação do subdomínio principal. Não traz nenhuma vantagem competitiva e a maioria das empresas possui uma implementação similar. Algumas soluções possíveis são:

Adquirir soluções existentes e disponíveis no mercado.

Internalizar o desenvolvimento e realizar através de uma equipe que não tem os recursos intelectuais mais importantes e custosos para a organização, que normalmente estão alocados no subdomínio principal.

Terceirizar o desenvolvimento.

Segundo Vaughn Vernon, os subdomínios estão no campo do problema, e os contextos delimitados no campo da solução.

Problema		**Solução**
Domínio	"é representado por"	Modelo de domínio
Subdomínio		Contexto delimitado

O que se espera é que exista um contexto delimitado para cada subdomínio, mas existem cenários onde pode haver um aprofundamento maior. Quando determinado subdomínio é muito grande, ele pode possuir mais de um contexto delimitado que faz parte de um todo maior.

Context Map (Mapa de contexto): Mapeia os contextos delimitados de um domínio, e descreve o relacionamento entre eles. Com o mapa de contexto, é possível entender a comunicação entre as equipes que desenvolvem cada contexto delimitado e saber se são mais ou menos colaborativas.

Segundo Vladik Khononov (Livro: What is Doman-Driven Design?),

> "os padrões de design orientado ao domínio servem para definir relacionamentos e integrações entre contextos delimitados. Esses padrões são impulsionados pela natureza da colaboração entre equipes que trabalham em contextos delimitados. Divide-se os padrões em três grupos, cada um representando um tipo de colaboração entre equipes: cooperation (cooperação), customer–supplier (cliente-fornecedor) e separate ways (caminhos separados)."
>
> Cooperation (Cooperação): Os padrões de cooperação referem-se a contextos delimitados implementados por equipes com comunicação bem estabelecida. Naturalmente, esse requisito é cumprido para contextos delimitados que são implementados pela mesma equipe. Também se aplica a equipes que têm objetivos dependentes, onde o sucesso de uma equipe depende da outra e vice-versa. Novamente, o principal critério aqui é a qualidade da comunicação e colaboração das equipes.

Vamos dar uma olhada em dois padrões DDD adequados para equipes cooperantes: Partnership (Parceria) e Shared kernel (Núcleo compartilhado).

Partnership (Parceria)

No modelo de parceria, a integração entre contextos delimitados é coordenada de forma ad hoc. Uma equipe pode notificar uma segunda sobre uma alteração na API, e a segunda equipe cooperar e se adaptar sem dramas ou conflitos.

A coordenação da integração aqui é de mão dupla. Nenhuma equipe dita a linguagem usada para definir os contratos. As equipes podem resolver as diferenças e escolher a solução mais adequada. Além disso, ambos os lados cooperam na solução de quaisquer problemas de integração que possam surgir. Nenhum dos dois times está interessado em bloquear o outro.

Práticas de colaboração bem estabelecidas, altos níveis de comprometimento e sincronizações frequentes entre as equipes são necessários para uma integração bem-sucedida dessa maneira.

Observe que esse padrão pode não ser adequado para equipes distribuídas geograficamente, uma vez que pode apresentar desafios de sincronização e comunicação.

Shared Kernel (Núcleo compartilhado)

O núcleo compartilhado é uma maneira mais formal de definir um contrato entre vários contextos delimitados. Aqui, em vez de integrações ad hoc, o contrato é definido explicitamente em uma biblioteca compilada e o núcleo compartilhado. A biblioteca define os métodos de integração e a linguagem usados por ambos os contextos delimitados.

O núcleo compartilhado é referenciado e de propriedade de vários contextos delimitados. Cada equipe é livre para modificar a biblioteca compilada que define o contrato de integração. Uma mudança no contrato pode quebrar a construção da outra equipe, no entanto; assim, como no caso da parceria, esse padrão exige altos níveis de comprometimento e sincronização entre as equipes.

Um detalhe peculiar sobre o padrão de núcleo compartilhado é que, de certa forma, ele contradiz um princípio central de contextos delimitados: que apenas uma equipe pode possuir um contexto delimitado. Aqui extraímos uma parte compartilhada de vários contextos delimitados em seu próprio contexto delimitado. Como resultado, o contexto delimitado compartilhado é de propriedade conjunta de várias equipes.

A chave para implementar o padrão de núcleo compartilhado é manter o escopo do núcleo compartilhado pequeno e limitado apenas ao contrato de integração entre contextos delimitados.

Customer–Supplier (Cliente-Fornecedor)

Aqui um dos contextos delimitados — o fornecedor — fornece um serviço para seus clientes. O provedor de serviços é "upstream" e o cliente ou consumidor é "downstream". O contexto fornecedor, fornece algo a outros contextos e pode-se dizer que ele mantém o que outros contextos necessitam.

Ao contrário do caso da cooperação, ambas as equipes podem ter sucesso de forma independente. Mas na maioria dos casos, temos um desequilíbrio de poder.

Discutiremos três padrões que abordam essas diferenças de poder: Conformista, camada anticorrupção e serviço de host aberto.

Conformist (Conformista)

> Em alguns casos, o equilíbrio de poder é a favor da equipe upstream, que não tem motivação real para apoiar as necessidades de seus clientes. Em vez disso, ela apenas fornece o contrato de integração, definido de acordo com seu próprio modelo, é pegar ou largar. Tais desequilíbrios de poder podem ser causados pela integração com prestadores de serviços externos à organização, ou simplesmente por políticas organizacionais.
>
> Se a equipe downstream puder aceitar o modelo da equipe upstream, a relação entre os contextos delimitados é chamada de conformista. Pois a equipe downstream está conformada com o modelo da equipe upstream.
>
> A decisão da equipe downstream de abrir mão de parte de sua autonomia pode ser justificada de várias maneiras. Por exemplo, o contrato exposto pela equipe de upstream pode ser um modelo padrão da indústria e bem estabelecido, ou pode ser bom o suficiente para as necessidades da equipe de downstream.

Anticorruption Layer (Camada Anticorrupção)

> Camada anticorrupção, como no caso do padrão conformista, o equilíbrio de poder nessa relação ainda está inclinado para o serviço upstream. No entanto, neste caso, o contexto delimitado downstream não está disposto a se conformar. O que ele pode fazer, em vez disso, é traduzir o modelo do contexto delimitado upstream em um modelo adaptado às suas próprias necessidades por meio de uma camada anticorrupção.

Open-Host Service (Serviço de host aberto)

Esse padrão aborda o caso em que a energia é enviesada para os consumidores. O fornecedor está interessado em proteger seus consumidores e prestar o melhor serviço possível.

Para proteger os consumidores de mudanças em sua implementação, o fornecedor upstream desacopla seu modelo de implementação da interface pública. Esse desacoplamento permite que o fornecedor evolua sua implementação e modelos públicos em diferentes taxas.

A interface pública do fornecedor não se destina a estar em conformidade com a sua linguagem ubíqua. Em vez disso, pretende-se expor um protocolo conveniente para os consumidores, expresso em uma linguagem orientada à integração. Por isso, o protocolo público é chamado de "linguagem publicada".

Em certo sentido, o padrão de serviço de host aberto é uma inversão do padrão da camada anticorrupção: em vez do consumidor, o fornecedor implementa a tradução de seu próprio modelo interno.

Separate Ways (Caminhos separados)

A última opção de colaboração é, claro, não colaborar. Esse padrão pode surgir por diferentes motivos, nos casos em que as equipes não estão dispostas ou aptas a colaborar. Veremos alguns motivos aqui.

Questões de Comunicação

Uma razão comum para evitar a colaboração são as dificuldades de comunicação motivadas pelo tamanho da organização ou por questões políticas internas. Quando as equipes têm dificuldade em colaborar e concordar, pode ser mais econômico para elas seguirem caminhos separados e duplicarem a funcionalidade em vários contextos delimitados.

Subdomínios genéricos

A natureza do subdomínio duplicado também pode ser um motivo para as equipes seguirem caminhos separados. Mais especificamente, quando o subdomínio em questão é genérico, se a solução genérica é fácil de integrar, pode ser mais econômico integrá-lo em cada um dos contextos delimitados localmente. Um exemplo é uma estrutura de log; faria pouco sentido para um dos contextos delimitados expô-lo como um serviço, pois a complexidade adicional da integração de tal solução superaria o benefício de não duplicar a funcionalidade em vários contextos. Duplicar a funcionalidade seria mais barato do que a colaboração.

Diferenças de modelo

A diferença nos modelos de contextos delimitados também pode ser um motivo para seguir caminhos separados. Os modelos podem ser tão diferentes que uma relação conformista não é possível, e implementar uma camada anticorrupção seria mais caro do que duplicar a funcionalidade. Nesse caso, novamente, é mais econômico para as equipes seguirem caminhos separados.

Quando evitar

O padrão de caminhos separados deve ser evitado ao integrar subdomínios principais. Duplicar a implementação de tais subdomínios desafiaria a estratégia da empresa de implementá-los da maneira mais eficaz e otimizada.