RESENHA DOS CAPÍTULOS IV, V E VI DO ARTIGO "DOMAIN-DRIVEN DESIGN REFERENCE"

Resenha da obra: EVANS, Eric. **Domain-Driven Design Reference: Definitions and Pattern Summaries**. Domain Language, Inc., 2015.

A presente resenha se presta a analisar os capítulos IV, V e VI do artigo "Domain-Driven Design Reference" (Referência de Design Orientado ao Domínio - DDD) de Eric Evans, que tratam do Design Estratégico.

O Design Estratégico em DDD lida com o projeto e a organização de grandes sistemas, focando em como dividir o sistema em partes coerentes e como essas partes se relacionam, com o objetivo de manter a clareza e o foco no **Core Domain** (Domínio Central).

IV. Context Mapping for Strategic Design

O Capítulo IV, **Context Mapping for Strategic Design** (Mapeamento de Contexto para Design Estratégico), aborda a complexidade de gerenciar múltiplos modelos em um grande projeto. Modelos distintos são inevitáveis, surgindo de diferentes comunidades de usuários, equipes independentes ou tecnologias. No entanto, a combinação de código baseado em modelos distintos torna o software propenso a falhas e confuso.

O padrão fundamental aqui é o **Bounded Context** (Contexto Delimitado), que é a descrição de um limite (como um subsistema ou o trabalho de uma equipe) dentro do qual um modelo particular é definido e aplicável. O **Context Map** (Mapa de Contexto) se torna essencial para fornecer uma visão realista e em grande escala de como os modelos se estendem pelo projeto e como se integram com outros. Ele deve identificar cada modelo e seu Contexto Delimitado, nomeá-los e descrever os pontos de contato e as relações entre eles.

O capítulo detalha nove padrões de relacionamento entre Contextos Delimitados, reconhecendo as complexidades de comunicação e dependência:

- **Partnership** (Parceria): Usado quando o fracasso de desenvolvimento em qualquer um dos dois contextos resultaria em fracasso de entrega para ambos (relação de dependência mútua). Exige planejamento coordenado e gestão conjunta da integração.
- **Shared Kernel** (Núcleo Compartilhado): Designa explicitamente um pequeno subconjunto do modelo de domínio e do código associado que as equipes concordam em compartilhar, exigindo um processo de integração contínua para manter o modelo alinhado.
- **Customer/Supplier Development** (Desenvolvimento Cliente/Fornecedor): Relação **upstream-downstream** (a montante-a jusante) onde as prioridades da equipe a jusante (cliente) são orçadas e negociadas no planejamento da equipe a montante (fornecedora). Testes de aceitação automatizados desenvolvidos em conjunto ajudam a proteger a interface a jusante.

- **Published Language** (Linguagem Publicada): Utiliza-se uma linguagem bem documentada e compartilhada (como um padrão de intercâmbio de dados) como meio comum de comunicação, traduzindo de e para essa linguagem.

- **Separate Ways** (Caminhos Separados): Declara-se que um Contexto Delimitado não tem conexão alguma com os outros, permitindo soluções simples e especializadas em um escopo pequeno.

- **Big Ball of Mud** (Grande Bola de Lama): Reconhece-se uma área onde os modelos estão misturados e os limites são inconsistentes. O conselho é traçar um limite ao redor da confusão e não tentar aplicar modelagem sofisticada.

V. Distillation for Strategic Design

O Capítulo V, **Distillation for Strategic Design** (Destilação para Design Estratégico), trata de focar no problema central e evitar ser inundado por problemas laterais. Destilação é o processo de separar componentes para extrair a essência, tornando-a mais valiosa e útil.

O conceito central é o **Core Domain** (Domínio Central), o coração do sistema, que deve ser o foco dos desenvolvedores mais talentosos e onde o modelo deve ser mais aprofundado e o design mais **Supple** (Flexível). Os padrões-chave são:

- **Core Domain**: Ferver o modelo, definindo um Domínio Central pequeno e fornecendo meios para distingui-lo do suporte.

- **Generic Subdomains** (Subdomínios Genéricos): Identificar subdomínios coesos que não são a motivação do projeto (por exemplo, sistemas de e-mail, gerenciamento de usuários). Fatorar modelos genéricos para esses subdomínios em módulos separados e considerá-los de menor prioridade ou soluções prontas para uso.

- **Domain Vision Statement** (Declaração de Visão de Domínio): Uma descrição curta (cerca de uma página) do Domínio Central e o valor que ele trará, focando nos aspectos que o distinguem.

- **Highlighted Core** (Núcleo Destacado): Documento breve (três a sete páginas esparsas) ou marcação dos elementos do Domínio Central no repositório do modelo, para tornar o núcleo mais fácil de identificar.

- **Cohesive Mechanisms** (Mecanismos Coesos): Particionar um mecanismo conceitualmente coeso (muitas vezes algoritmos complexos) em um **framework** separado e leve, expondo suas capacidades com uma interface que revela a intenção. Isso permite que o resto do domínio se concentre no o quê" (o problema), delegando as complexidades da solução (o como").

- **Segregated Core** (Núcleo Segregado): Refatorar o modelo para separar os conceitos centrais dos coadjuvantes (incluindo os genéricos) em outros pacotes, fortalecendo a coesão do núcleo e reduzindo seu acoplamento com outro código.

- **Abstract Core** (Núcleo Abstrato): Identificar os conceitos diferenciadores mais fundamentais, fatorando-os em classes abstratas ou interfaces. Este modelo abstrato é projetado para expressar a maior parte da interação entre componentes significativos e é colocado em seu próprio módulo, separadamente das classes de implementação detalhada.

VI. Large-scale Structure for Strategic Design

O Capítulo VI, **Large-scale Structure for Strategic Design** (Estrutura em Grande Escala para Design Estratégico), trata da necessidade de um princípio abrangente para interpretar os elementos de um grande sistema em termos de seu papel no todo.

Uma "estrutura em grande escala" é uma linguagem que permite entender o sistema em linhas gerais, um padrão de regras, papéis e relacionamentos que abrange todo o sistema para guiar o design e auxiliar a compreensão. A principal recomendação é permitir que essa estrutura conceitual evolua com a aplicação, ajustando-se ao domínio, sem restringir excessivamente as decisões de design detalhado.

Os padrões específicos de estrutura em grande escala apresentados incluem:

- **System Metaphor** (Metáfora do Sistema): Adotar uma analogia concreta para organizar o design e absorvê-la na linguagem onipresente, facilitando a comunicação e guiando o desenvolvimento.

- **Responsibility Layers** (Camadas de Responsabilidade): Olhar para as dependências conceituais e as diferentes taxas de mudança no domínio, definindo estratos naturais como amplas responsabilidades abstratas (camadas). O modelo deve ser refatorado para que as responsabilidades de cada objeto se encaixem perfeitamente na responsabilidade de uma camada.

- **Knowledge Level** (Nível de Conhecimento): Criar um conjunto distinto de objetos que descrevem e restringem a estrutura e o comportamento do modelo básico. Isso separa as preocupações em dois "níveis": um concreto e outro refletindo regras e conhecimentos que um usuário ou superusuário pode customizar.

- **Pluggable Component Framework** (Framework de Componentes Conectáveis): Destilar um núcleo abstrato de interfaces e interações, criando um **framework** que permite que implementações diversas dessas interfaces sejam substituídas livremente. Isso é geralmente aplicável a um modelo muito maduro, após algumas aplicações terem sido implementadas no mesmo domínio.

Os três capítulos fornecem um arcabouço para o **Strategic Design** em DDD. O **Mapeamento de Contexto** define as fronteiras e os relacionamentos, a **Destilação** garante que o foco seja mantido no Domínio Central e a **Estrutura em Grande Escala** fornece os princípios organizacionais para dar coerência ao sistema como um todo. Juntos, eles permitem que sistemas complexos sejam desenvolvidos com clareza conceitual e facilidade de manutenção.