Resenha - Domain-Driven Design Reference

Capítulos 4, 5 e 6

Autor: Estudante de Engenharia de Software

Data: 29/09/2025

Esta resenha apresenta uma visão prática dos capítulos 4, 5 e 6 do Domain-Driven Design Reference (Eric Evans, 2015), com foco no que um(a) estudante de Engenharia de Software pode aplicar no dia a dia: mapear contextos, destilar o domínio e dar estrutura em larga escala sem engessar a evolução.

Capítulo 4 - Context Mapping (Mapeamento de Contextos)

O Context Map torna explícitas as fronteiras entre bounded contexts e, principalmente, o tipo de relação entre eles. Nomear essas relações reduz ambiguidade e ajuda times a negociar integrações.

- **Partnership**: dois times com dependência forte decidem juntos aspectos de domínio.
- **Shared Kernel**: compartilhar um pequeno núcleo de modelo/código; útil, mas mantenha pequeno para evitar acoplamento excessivo.
- **Customer/Supplier Development**: o upstream (supplier) negocia interfaces para dar previsibilidade ao downstream (customer).
- **Conformist**: o downstream adota o modelo do upstream quando não há poder de barganha - decisão consciente.
- **Anticorruption Layer (ACL)**: camada de tradução para proteger seu modelo de legados/externos.
- **Open-host Service**: expor um protocolo/serviço estável a múltiplos consumidores distintos.
- **Published Language**: padronizar a linguagem/dados de integração (pode combinar com open-host).
- **Separate Ways**: quando integrar custa mais do que beneficia, separe e deixe cada contexto seguir.
- **Big Ball of Mud** (alerta): sem fronteiras claras, o sistema vira uma massa amorfa.

Por que importa? Sem um Context Map vivo e nomes claros para as relações, o acoplamento cresce silenciosamente e a coordenação entre equipes degrada.

Capítulo 5 - Distillation (Destilação do Domínio)

Destilar é concentrar o esforço no que é o coração do negócio (Core Domain) e isolar o restante. O capítulo oferece um kit de padrões para destacar o que realmente cria vantagem competitiva.

- **Core Domain**: identificar e manter pequeno o núcleo que diferencia o produto; priorize e aloque as melhores pessoas.
- **Generic Subdomains**: o que não é core, mas é necessário; isole e considere soluções de prateleira.
- **Domain Vision Statement**: enunciado curto que explica valor e foco do sistema.
- **Highlighted Core**: tornar o core visível e rastreável (documento sucinto e marcações no repositório).
- **Cohesive Mechanisms**: encapsular mecanismos/algoritmos complexos com interfaces claras.
- **Segregated Core**: separar estruturalmente conceitos core de elementos de suporte.
- **Abstract Core**: visão mais abstrata para comunicar o panorama quando há muitos detalhes e interações.

Resumo prático: ferver o modelo até sobrar o essencial; isolar o resto para que não engula o core.

Capítulo 6 - Large-scale Structure (Estrutura em Larga Escala)

Estruturas macro dão coerência ao sistema sem engessá-lo. O objetivo é oferecer ordem suficiente para enxergar o todo, preservando a autonomia local para decisões de detalhe.

- **Evolving Order**: evitar tanto a anarquia quanto a camisa de força; a estrutura de alto nível deve evoluir com o produto.
- **System Metaphor**: metáfora compartilhada para orientar design e comunicação (reavalie periodicamente).
- **Responsibility Layers**: camadas organizadas por responsabilidade acima do modelo.
- **Knowledge Level**: separar objetos que descrevem/restringem o comportamento do modelo base (regras configuráveis).
- **Pluggable Component Framework**: em domínios maduros, extrair um framework de componentes plugáveis para interoperabilidade.

Mensagem central: dê ordem macro suficiente para enxergar o todo, sem travar a evolução e o aprendizado do domínio.

Como os capítulos se complementam

O Cap. 4 define fronteiras e relações entre modelos; o Cap. 5 destila o que importa; e o Cap. 6 dá forma em larga escala. Juntos, evitam o Big Ball of Mud e ajudam a transformar o modelo de domínio em um ativo estratégico do produto.

Notas práticas de um(a) estudante

- Crie e mantenha vivo um Context Map no repositório e nos PRDs.
- Proteja seu BC com ACL ao integrar com legados; adote Conformist quando necessário, de forma consciente.
- Invista no Core Domain: mantenha pequeno, visível e com time sênior.
- Isole Generic Subdomains e encapsule mecanismos complexos como componentes coesos.
- Dê uma estrutura macro evolutiva e reavalie metáforas/layers a cada ciclo.