Resenha Capítulos 4,5 e 6

Strategic Design and Domain-Driven Design

Diogo Brunoro

September 2025

Esta resenha apresenta uma síntese dos capítulos 4, 5 e 6 do livro *Domain-Driven Design Reference*, de Eric Evans (2015). Os capítulos exploram aspectos estratégicos do DDD, focando em como mapear contextos, destilar o domínio e estruturar sistemas em grande escala. A análise é feita a partir da perspectiva de um estudante universitário, buscando destacar as principais ideias e impressões.

No capítulo 4, Evans discute a importância do *Context Mapping*, que consiste em entender como diferentes *Bounded Contexts* se relacionam dentro de um sistema complexo. Ele mostra que não existe um único modelo que atenda a todas as partes do software: vários modelos surgem e precisam coexistir. São apresentados padrões de relacionamento como *Parceria*, *Shared Kernel*, *Customer/Supplier*, *Conformist* e o *Anticorruption Layer*, que ajudam a evitar dependências problemáticas. A sensação é que este capítulo funciona como um "mapa político" do software, em que definir fronteiras e alinhar linguagens é tão importante quanto escrever código.

Aideia central do capítulo 5 é *destilar o domínio* para encontrar o que realmente importa, o chamado *Core Domain*. Evans explica que todo sistema possui partes genéricas, que podem até ser terceirizadas, e partes que representam o verdadeiro diferencial do negócio. O desafio é identificar esse "coração" e concentrar esforços nele, evitando desperdício de tempo em áreas sem valor estratégico. O autor apresenta padrões como *Generic Subdomains*, *Domain Vision Statement* e *Highlighted Core*, que ajudam a manter o foco. Este capítulo faz o leitor pensar como um designer de produto, e não apenas como programador.

O último capítulo amplia a visão para a *estrutura em larga escala* do sistema. Evans propõe maneiras de organizar a arquitetura para que ela continue compreensível e flexível à medida que o software cresce. São explorados conceitos como *Evolving Order*, *System Metaphor*, *Responsibility Layers*, *Knowledge Level* e *Pluggable Component Framework*. A ideia é criar "esqueletos" que guiem a evolução do sistema sem engessá-lo, permitindo adaptação constante. A arquitetura é tratada como algo vivo, que precisa de regras claras para não se transformar em um *Big Ball of Mud*.

Os três capítulos analisados demonstram que o *Domain-Driven Design* vai muito além da modelagem de classes e objetos. Eles evidenciam a necessidade de pensar em fronteiras, prioridades e estruturas de longo prazo para lidar com a complexidade dos sistemas. Para um estudante de tecnologia, a leitura é

um convite a enxergar o desenvolvimento de software como uma atividade estratégica, que envolve negociação, foco no negócio e planejamento arquitetural.