DDD Reference

Capítulo IV – Context Mapping for Strategic Design

Este capítulo trata da importância de entender e mapear os diferentes Bounded Contexts dentro de um sistema complexo. Como grandes sistemas raramente têm um único modelo, é essencial definir como esses contextos se relacionam. Evans apresenta padrões estratégicos para lidar com essas interações:

- Parcerias e Shared Kernel: colaboração próxima e compartilhamento de partes do modelo.
- Customer/Supplier e Conformist: quando um contexto depende das decisões de outro, podendo adotar ou adaptar modelos.
- Anticorruption Layer e Published Language: maneiras de proteger a integridade de um contexto frente a outro, estabelecendo contratos claros.
- Separate Ways e Big Ball of Mud: reconhecer quando separar contextos é melhor do que tentar integrá-los.

→ Ideia central: mapear explicitamente os relacionamentos entre contextos evita ambiguidades, falhas de comunicação e reduz riscos de “acoplamento caótico”.

Capítulo V – Distillation for Strategic Design

Neste capítulo, Evans mostra como destilar o domínio para destacar o que é realmente essencial ao negócio. O objetivo é identificar o Core Domain (o núcleo estratégico) e separá-lo de partes mais genéricas ou de suporte.

- Core Domain: onde está o maior valor competitivo do sistema.
- Generic Subdomains: funcionalidades que podem ser reutilizadas ou adquiridas, sem valor estratégico único.
- Domain Vision Statement e Highlighted Core: formas de manter a visão clara do que é central.
- Cohesive Mechanisms e Segregated Core: técnicas para manter o núcleo coeso e protegido.

- Abstract Core: busca de modelos conceituais mais profundos que sintetizam o domínio.

→ Ideia central: concentrar esforços de design no núcleo do negócio, delegando ou simplificando o que não é diferencial. Isso garante foco estratégico e melhor aproveitamento de recursos.

Capítulo VI – Large-scale Structure for Strategic Design

Aqui, o autor discute estruturas que ajudam a organizar sistemas complexos em larga escala, mantendo coerência e clareza ao longo do tempo.

- Evolving Order: aceitar que a estrutura vai emergir e evoluir com o aprendizado.
- System Metaphor: uso de metáforas para criar uma visão compartilhada da arquitetura.
- Responsibility Layers: dividir responsabilidades em camadas, mantendo separação clara.
- Knowledge Level: separar regras de negócio mais estáveis de regras que mudam com frequência.
- Pluggable Component Framework: desenhar componentes intercambiáveis para flexibilidade.

→ Ideia central: em projetos grandes, a organização estrutural deve guiar equipes e decisões, permitindo evolução contínua sem perder clareza.