ANÁLISE DO ARTIGO
“STRATEGIC DESIGN AND DOMAIN-DRIVEN DESIGN”
VINÍCIUS GOMES RODRIGUES
ENGENHARIA DE SOFTWARE - PUC MINAS
BELO HORIZONTE - 2025

O texto apresenta ferramentas de alto nível (Design Estratégico) para gerenciar a complexidade em sistemas grandes e descentralizados, onde múltiplos modelos e equipes estão envolvidos. O objetivo é criar um sistema coeso e bem-integrado, mesmo com partes independentes.

Garlan estrutura seu argumento de forma linear e evolutiva, dividindo-o em três eixos temporais:

1. Context Mapping (Mapa de Contexto)

O foco é entender e definir claramente as fronteiras (Bounded Contexts) onde um modelo específico é aplicável. O ponto central é mapear os relacionamentos entre esses contextos. O texto apresenta um conjunto de padrões para definir esses relacionamentos:

- Partnership (Parceria): Duas equipes com sucesso mutuamente dependente cooperam diretamente.
- Shared Kernel (Núcleo Compartilhado): Uma pequena parte do modelo e código é compartilhada e integrada continuamente por duas equipes.
- Customer/Supplier (Cliente/Fornecedor): Relacionamento upstream/downstream formal, onde o time "cliente" (downstream) tem suas necessidades consideradas no planejamento do "fornecedor" (upstream).
- Conformist (Conformista): O time downstream se conforma ao modelo upstream, simplificando a integração à custa de um modelo próprio ideal.
- Anticorruption Layer (Camada Anticorrupção): O time downstream cria uma camada de tradução para isolar e proteger seu modelo do modelo "corrompido" ou complexo de um upstream.
- Open-Host Service (Serviço de Host Aberto): Um contexto upstream define um protocolo/service bem definido para que vários downstreams possam se integrar.
- Published Language (Linguagem Publicada): Uma linguagem bem-documentada (como um padrão de XML ou JSON) usada para a comunicação entre contextos, frequentemente associada ao Open-Host.

- Separate Ways (Caminhos Separados): Reconhece que a integração tem um custo e, se o benefício for pequeno, é melhor deixar os contextos totalmente independentes.
- Big Ball of Mud (Grande Bola de Lama): Um padrão para reconhecer e delimitar áreas do sistema que são uma bagunça irremediável, contendo-as para que não "vazem" para outros contextos.

2. Distillation (Destilação)

Preocupado em como destacar e priorizar o Domínio Central (Core Domain) – a parte mais valiosa e diferenciada do software – em meio a uma grande quantidade de código e funcionalidades necessárias, mas genéricas.

- Core Domain: A parte mais crítica do modelo, que deve receber o maior talento e esforço de design.
- Generic Subdomains (Subdomínios Genéricos): Partes do sistema importantes, mas não críticas ou diferenciadas (ex.: biblioteca de e-mail, módulo de logging). Devem ser segregadas e, se possível, implementadas com soluções de prateleira.
- Domain Vision Statement: Uma declaração de uma página que define o valor e o foco do Core Domain.
- Highlighted Core (Núcleo Destacado): Técnicas para deixar o Core Domain visível no código (documentação, marcações, etc.).
- Cohesive Mechanisms (Mecanismos Coesos): Isolar algoritmos complexos em frameworks para que o modelo de domínio possa se focar na "regra de negócio" (o "o quê"), não na complexidade implementação (o "como").
- Segregated Core (Núcleo Segregado): Refatorar o código para separar fisicamente os elementos do Core Domain dos elementos de suporte, melhorando a coesão e reduzindo o acoplamento.
- Abstract Core (Núcleo Abstrato): Criar uma camada de abstrações (classes, interfaces) que capturem os conceitos e interações mais essenciais do domínio, servindo como a espinha dorsal do modelo.

3. Large-Scale Structure (Estrutura em Grande Escala)

Define princípios arquiteturais de alto nível que dão ordem e coerência ao sistema como um todo, guiando o design e facilitando o entendimento.

- Evolving Order (Ordem Evolutiva): A estrutura de grande escala deve evoluir com o projeto, não ser um arcabouço rígido definido upfront.
- System Metaphor (Metáfora do Sistema): Usar uma analogia concreta (ex.: "uma linha de montagem", "um leilão") para guiar o design e facilitar a comunicação. Deve ser usada com cuidado, pois metáforas são imperfeitas.
- Responsibility Layers (Camadas de Responsabilidade): Organizar o modelo em camadas conceituais amplas (ex.: "Operações", "Estratégia", "Política") baseadas em taxas de mudança e dependências.
- Knowledge Level (Nível de Conhecimento): Uma camada de metadados que descreve e configura o comportamento de outra camada (o "Nível Operacional"). Permite alta customização sem alterar o código base.
- Pluggable Component Framework (Framework de Componente Plugável): O ponto mais alto de maturidade, onde um núcleo abstrato de interfaces permite que diversos componentes concretos sejam plugados e interoperem.

O texto apresenta os pilares do Design Estratégico em Domain-Driven Design (DDD), fornecendo um framework para gerenciar a complexidade em sistemas grandes com múltiplas equipes e modelos. Ele transcende a modelagem tática, posicionando as decisões de arquitetura como um elemento estratégico para o alinhamento de times, a gestão de dependências e a maximização do valor do domínio de negócio.

Através de conceitos como Context Mapping (mapeamento de relacionamentos entre bounded contexts), Distillation (foco no Core Domain) e Large-Scale Structure (estruturas de alto nível para coerência global), o artigo oferece um vocabulário prático para transformar dinâmicas sociais e de negócio em padrões de design eficazes. Sua abordagem é notavelmente pragmática, reconhecendo realidades como a existência de sistemas legados caóticos (Big Ball of Mud) e a necessidade de evolução contínua da arquitetura (Evolving Order).