Aluno: Vitor Veiga Silva

Strategic Design and Domain-Driven Design

O Design Orientado ao Domínio (DDD) é uma maneira de criar softwares complexos que procura harmonizar o código com o saber específico da área de atuação da empresa. A ideia principal é ter o modelo do domínio como o centro do sistema, criado em conjunto por quem entende do negócio e por quem desenvolve, usando uma linguagem clara que evite dúvidas e ajude na comunicação.

Entende-se por domínio o campo de conhecimento ou a atividade onde o software opera, e o modelo é um resumo que mostra os pontos importantes desse domínio, permitindo solucionar problemas de um jeito mais fácil. Para isso, é essencial usar a chamada linguagem ubíqua, um conjunto de termos usado por toda a equipe técnica e pelos especialistas do negócio, que deve estar presente tanto no código quanto nas conversas e nos documentos. Essa linguagem vale dentro de um contexto específico, ou seja, uma fronteira que define onde um certo modelo faz sentido, evitando confusão e interpretações erradas.

O DDD também prioriza práticas que mantêm o sistema consistente. A integração contínua é uma delas, pois garante que todos os desenvolvedores mantenham o modelo unificado dentro de um mesmo contexto. O design guiado pelo modelo busca garantir que o código reflita diretamente o modelo do domínio, de modo que qualquer mudança no código também represente uma evolução na forma como entendemos o modelo. A refatoração é vista não só como uma melhoria técnica, mas também como um jeito de entender melhor o domínio. Outro ponto importante é a participação de quem cria o modelo, já que

desenvolvedores e especialistas devem trabalhar juntos, mantendo o modelo sempre ligado à realidade do negócio.

Entre as peças que formam o DDD estão a arquitetura em camadas, que separa a lógica do domínio da infraestrutura e da interface, as entidades, que têm sua própria identidade ao longo do tempo, e os objetos de valor, definidos apenas pelo que têm. Também existem os eventos de domínio, que mostram acontecimentos importantes, os serviços, que juntam operações que não pertencem a uma entidade específica, os módulos, que organizam ideias relacionadas, e os agregados, que unem entidades e objetos de valor em torno de uma raiz. Para ajudar a guardar e criar objetos, surgem os repositórios e as fábricas.

Além das peças básicas, o DDD propõe práticas de design que deixam o modelo mais claro e fácil de mudar, como interfaces que mostram as intenções, funções sem efeitos colaterais, uso de classes independentes, fechamento de operações e design declarativo. Em um nível mais amplo, também existem os mapas de contexto, que mostram as relações entre diferentes modelos em sistemas grandes, incluindo padrões como parceria, núcleo compartilhado, fornecedor/cliente, conformista, camada anticorrupção, linguagem publicada e caminhos separados.

Um aspecto crucial é refinar o modelo, visando realçar a essência do campo de atuação, apartando áreas secundárias e ferramentas de apoio. A meta é assegurar a nitidez do que é vital para a empresa e garantir que os esforços da equipe se concentrem onde o impacto é maior. Em projetos extensos, modelos de arquitetura de grande escala promovem a consistência, como divisões de tarefas, analogias de sistema, graus de especialização e plataformas de elementos integráveis.

Além disso, é fundamental frisar que o DDD não se resume a abordagens de modelagem ou esquemas de projeto, mas afeta também a maneira

como os times se organizam e se relacionam. A precisão na comunicação, o foco no domínio e a interação constante entre setores técnicos e de negócio estabelecem um cenário onde o software não só satisfaz as demandas atuais, mas também acompanha o progresso do saber adquirido com o tempo.

Assim, o DDD deve ser encarado como uma doutrina de desenvolvimento que mescla estratégia e execução, unindo teoria e prática, e viabilizando a construção de sistemas intricados de modo mais coeso, transparente e duradouro, sempre espelhando de forma autêntica o cenário empresarial que representam.