Nos capítulos 4, 5 e 6 da obra *Strategic Design and Domain-Driven Design*, o autor concentra-se nos elementos táticos que dão sustentação ao design estratégico, destacando padrões de modelagem que permitem articular a lógica de negócio com clareza conceitual e robustez técnica. Esses capítulos não apenas descrevem estruturas e mecanismos, mas também discutem suas implicações para a manutenção da integridade do domínio e para a evolução da arquitetura de software em contextos complexos.

O capítulo 4 dedica-se ao estudo dos repositórios e da unidade de trabalho, enfatizando sua função como pontes entre o modelo de domínio e a camada de infraestrutura. Os repositórios são definidos como abstrações que simulam coleções em memória, mas que, na prática, encapsulam operações de persistência, proporcionando ao desenvolvedor uma interface limpa para adicionar, remover ou buscar entidades. Esse mecanismo contribui para manter a lógica de domínio livre de dependências diretas com tecnologias específicas de armazenamento, favorecendo a independência do modelo e a possibilidade de substituição de implementações sem impacto conceitual. Ao mesmo tempo, o autor alerta para o risco de inflacionar repositórios com consultas arbitrárias, lembrando que seu papel não é substituir serviços especializados de busca, mas preservar as invariantes e a integridade do agregado ao qual estão vinculados. Em paralelo, a unidade de trabalho surge como um padrão complementar, responsável por tratar as alterações realizadas no modelo como parte de uma única transação. Esse recurso é fundamental para garantir atomicidade: ou todas as modificações são confirmadas, ou nenhuma delas é persistida. Assim, o capítulo evidencia como esses dois padrões, quando combinados, reforçam a consistência interna do domínio e promovem uma clara separação entre preocupações de negócio e aspectos técnicos de persistência.

No capítulo 5, o texto aprofunda-se nos padrões de design tático que estruturam o modelo de domínio. O ponto de partida são as entidades, caracterizadas por sua identidade única e duradoura, que lhes permite ser distinguidas mesmo diante de alterações em seus atributos. Em contraste, os objetos de valor são descritos como elementos sem identidade própria, definidos apenas por suas características e, por isso, frequentemente imutáveis. A distinção entre esses dois tipos de componentes é essencial para um modelo expressivo, já que evita o excesso de entidades desnecessárias e mantém o foco naquilo que de fato exige rastreabilidade ao longo do tempo. A discussão avança para os agregados e suas raízes, que representam grupos coesos de entidades e objetos de valor, dentro dos quais as regras de consistência são rigorosamente aplicadas. O princípio de acesso exclusivo à raiz do agregado assegura que a integridade interna seja preservada, evitando que elementos internos sejam manipulados de forma inconsistente. Além desses elementos, o autor descreve os serviços de domínio, operações que, embora cruciais, não se encaixam de maneira natural em entidades ou objetos de valor. Essa categoria de componentes permite que a lógica de negócio seja mantida explícita e independente de infraestrutura, preservando a clareza conceitual do modelo. Por fim, as fábricas são apresentadas como mecanismos de criação responsáveis por garantir que entidades e agregados sejam instanciados em estado válido, respeitando invariantes desde sua origem. Assim, o capítulo 5 consolida uma visão integrada dos padrões táticos, mostrando como cada um deles contribui para que o modelo seja robusto, claro e alinhado à linguagem ubíqua do domínio.

O capítulo 6, por sua vez, aprofunda-se especificamente nos serviços de domínio e nas fábricas, detalhando seus papéis, fronteiras e boas práticas de aplicação. No caso dos serviços, o autor ressalta que eles devem ser utilizados para representar operações que não pertencem a nenhuma entidade em particular, mas que ainda assim fazem parte central da lógica de negócio. Através deles, torna-se possível articular cálculos, processos e regras que atravessam múltiplos agregados ou que exigem a orquestração de diferentes entidades. Contudo, o autor alerta para o risco de transformar serviços em recipientes genéricos de lógica dispersa, o que levaria à perda de coesão e à diluição da expressividade do modelo. Os serviços de domínio, portanto, devem ser precisos, finos e semanticamente alinhados ao vocabulário do negócio. Quanto às fábricas, o capítulo destaca sua importância para a criação de estruturas complexas, como agregados compostos de múltiplas entidades e objetos de valor. Sua função primordial é assegurar que o processo de instanciamento respeite as invariantes, evitando que objetos sejam criados em estados inválidos ou inconsistentes. O autor diferencia fábricas simples, implementadas por meio de métodos estáticos, de fábricas mais elaboradas, organizadas como classes dedicadas, especialmente quando a complexidade do processo de criação assim o exige. Ao mesmo tempo, são apontados riscos, como o de transformar fábricas em meros instrumentos redundantes, desprovidos de lógica significativa. Por essa razão, a aplicação desse padrão deve ser criteriosa e guiada pela necessidade de preservar a integridade do domínio. Tanto serviços quanto fábricas se articulam diretamente com repositórios e unidades de trabalho, compondo um ecossistema de padrões que, em conjunto, sustentam a coesão do modelo e sua independência da infraestrutura técnica.

De maneira geral, os três capítulos evidenciam como os padrões táticos de DDD operam em sinergia para enfrentar a complexidade inerente a sistemas de software. A combinação de repositórios, unidades de trabalho, entidades, objetos de valor, agregados, serviços e fábricas permite que o modelo seja simultaneamente expressivo, consistente e resiliente às mudanças. Essa abordagem sustenta a ideia central do design orientado a domínio: a de que a arquitetura de software deve ser guiada pelo conhecimento profundo do negócio, e não pelas limitações da infraestrutura. Ao aplicar conscientemente esses padrões, o desenvolvedor não apenas organiza o código de forma mais clara, mas também contribui para a longevidade conceitual do sistema, permitindo que ele evolua sem comprometer sua integridade. Assim, os capítulos 4, 5 e 6 reforçam a noção de que o design estratégico é indissociável de uma base tática sólida, capaz de traduzir o domínio em estruturas de software coerentes e sustentáveis.