ARTIGO 8 - Managing Technical Debt

Leonardo Viana

O artigo “Managing Technical Debt”, de Steve McConnell, fala de um tema que todo desenvolvedor já enfrentou na prática: aquele dilema entre fazer algo rápido ou fazer algo direito. O autor compara muito bem a ideia de “dívida técnica” com a dívida financeira — você pega um atalho agora, mas vai ter que pagar juros mais tarde. E esses “juros” aparecem em forma de retrabalho, código difícil de manter, correções demoradas e até perda de velocidade no desenvolvimento.

Uma das partes que mais gostei foi a explicação sobre os dois tipos principais de dívida técnica: a não intencional, que surge por erro ou falta de experiência, e a intencional, que é quando a equipe decide conscientemente cortar caminho para ganhar tempo. Achei legal como o texto não demoniza totalmente a dívida técnica. Em alguns momentos ela é até estratégica, como em startups que precisam lançar rápido para não perder mercado. O problema está em não controlar ou não planejar o pagamento dessa dívida depois.

Outro ponto interessante é a divisão entre dívida de curto prazo e de longo prazo. A de curto prazo é aquela que precisa ser paga logo na próxima versão, como um empréstimo rápido. Já a de longo prazo é mais como um financiamento: você assume sabendo que vai carregar por um tempo. Essa comparação deixa o conceito muito mais fácil de entender, até para quem não é da área técnica, e mostra porque o artigo insiste tanto na importância de dar transparência para a dívida técnica dentro da empresa.

Gostei bastante da parte que fala sobre como comunicar a dívida técnica para pessoas de negócios. Muitas vezes gestores não entendem por que o time de desenvolvimento está pedindo tempo para “arrumar coisas” em vez de criar novas funcionalidades. O autor sugere traduzir o papo técnico para linguagem financeira, tipo dizer que 40% do orçamento está indo para “pagar juros” de soluções rápidas feitas no passado. Essa ideia é ótima porque aproxima mundos que costumam se chocar: o técnico e o administrativo.

Outro ponto marcante foi quando ele alerta contra o acúmulo de pequenas dívidas, o que ele chama de “crédito no cartão”. São aqueles detalhes que parecem bobos — variáveis mal nomeadas, código sem testes, classes mal divididas — mas que juntos viram uma bola de neve difícil de pagar. Achei esse trecho bem realista, porque mostra como muitas vezes o problema não é uma grande decisão errada, mas sim várias pequenas concessões que se somam.

O artigo também fala sobre estratégias de pagamento da dívida. Gostei do conselho de não tentar resolver tudo de uma vez com um mega projeto de “refatoração geral”. Na prática, isso quase nunca dá certo. É mais eficiente ir pagando aos poucos, inserindo parte do esforço de redução da dívida dentro do trabalho do dia a dia. Esse equilíbrio parece ser o caminho mais sustentável.

No geral, o que mais me chamou a atenção foi como o autor consegue transformar um tema que poderia parecer abstrato em algo bem concreto, cheio de exemplos e comparações simples. Ele mostra que a dívida técnica não é necessariamente um vilão, mas precisa ser tratada com responsabilidade, exatamente como uma dívida financeira. O que fica do artigo é que o importante não é nunca se endividar, mas saber quando vale a pena assumir a dívida e ter clareza sobre como e quando ela será paga.

Minha opinião é que esse texto é muito útil não só para desenvolvedores, mas também para gestores, justamente porque traduz uma dor muito comum das equipes de tecnologia. O maior aprendizado que levo é a importância de não varrer a sujeira para debaixo do tapete, mas sim tornar a dívida técnica visível, rastreável e comunicada de forma que todos entendam o impacto. E isso, para mim, é o ponto mais forte do artigo: ele mostra que com transparência e boa gestão, até as dívidas podem virar aliadas no crescimento de um projeto.