Resumo do Artigo

O artigo, escrito pelo Steve McConnell da Construx, é basicamente um guia sobre como lidar com a tal **Dívida Técnica**. O conceito foi criado pelo Ward Cunningham para descrever o *custo* que a gente assume no futuro quando escolhe uma solução rápida ou fácil agora, mas que aumenta a complexidade do sistema a longo prazo.

Assim como na economia, existem **dois tipos principais de Dívida Técnica**:

- **Dívida Não Intencional (Tipo I):** Essa é a *dívida ruim*. A gente cria sem querer, por falta de experiência, por fazer um trabalho de **baixa qualidade**, ou quando um *design* se mostra ruim e cheio de erros. O artigo diz que quanto menos desse tipo a equipe tiver, mais **Dívida Intencional** ela pode absorver de forma

segura.

- **Dívida Intencional (Tipo II):** Essa é a dívida que a gente *escolhe* assumir, o que pode ser uma decisão estratégica de negócios, como priorizar o **tempo de colocação no mercado** (*Time to Market*).

A **Dívida Intencional** ainda se divide em:

- **Curto Prazo (Tipo II.A):** Assumida reativamente, taticamente, geralmente para terminar uma *release* específica.
- **Focada (Tipo II.A.1):** Atalhos **rastreáveis e grandes**, como "hackear" algo para *shippar* e consertar depois. É comparada a um **empréstimo/ financiamento de carro**. **É um tipo saudável.**
- **Não Focada (Tipo II.A.2): Inúmeros atalhos minúsculos** (variáveis genéricas, comentários esparsos, falta de convenção). Acumula rápido, é

difícil de rastrear, e é comparada a um **dívida de cartão de crédito**. O artigo diz que esse tipo *não* vale a pena nem a curto prazo e deve ser evitado.

do.

- **Longo Prazo (Tipo II.B):** Assumida proativamente, estrategicamente, como um investimento (ex: não planejar o suporte a uma segunda plataforma pelos próximos 5 anos). Pode ser carregada por mais tempo. **É um tipo saudável.**

O Grande Problema: O Custo dos "Juros"

O ponto chave é que a Dívida Técnica precisa de **serviço**, ou seja, ela tem **juros**. Esses juros são o tempo extra gasto para manter o código funcionando ou a dificuldade de adicionar novas *features*

por causa da bagunça acumulada. Se a dívida cresce muito, você gasta mais para "serviçar" do que para inovar. Isso diminui a **velocidade** da equipe.
O artigo sugere aumentar a **transparência** da dívida, tratando-a como itens que precisam ser rastreados, seja em um sistema de *bug tracking* ou no *Scrum Product Backlog*, com esforço e prazo estimados, tal como um *story*.

Tomada de Decisão: Indo Além de Duas Opções

O artigo dá um conselho de ouro sobre a tomada de decisão quando estamos em cima do prazo: **não simplificar a escolha para apenas duas opções** (o "caminho bom e caro" vs. o "caminho rápido e sujo"). Ao invés disso, devemos considerar **três questões**:

- **Custo de Backfill:** Quanto custará para implementar o "caminho bom" *depois* de já ter implementado o "caminho rápido"? Inclui arrancar o código *quick and dirty* + implementar o código bom + custo do *quick path* original. O custo total será maior do que só fazer o caminho bom no início.
- **Pagamento de Juros:** Quanto o "caminho rápido" vai atrasar outros trabalhos até você conseguir refatorar? Isso pode ser um custo contínuo.
- **Terceira Opção:** Existe um caminho que é **rápido, mas não é sujo** (*Quick but not Dirty*)? Ou seja, um caminho que pode ser isolado do resto do sistema e que **não cria juros contínuos**?.

O artigo sugere que essa **terceira opção** (rápido, mas contido) é geralmente o **melhor caminho**, porque permite que você atinja o prazo no curto prazo e adie a

decisão de implementar o caminho bom *sem* ser forçado por um "juro" contínuo