ANÁLISE DO ARTIGO
“DOCUMENTING ARCHITECTURE DECISIONS”
VINÍCIUS GOMES RODRIGUES
ENGENHARIA DE SOFTWARE - PUC MINAS
BELO HORIZONTE - 2025

O artigo, escrito no formato de uma própria Architecture Decision Record (ADR), tem como objetivo principal introduzir e defender a adoção de ADRs como um método eficaz para documentar decisões arquiteturalmente significativas em projetos ágeis. Seu propósito central é resolver um problema crônico no desenvolvimento de software: a perda do contexto e da motivação por trás de decisões técnicas importantes ao longo do tempo, especialmente em equipes com rotatividade. O autor propõe uma alternativa prática à documentação tradicional, pesada e desatualizada, argumentando que a agilidade não é incompatível com a documentação, mas sim com a documentação sem valor.

O artigo estrutura o conceito de ADR de forma muito clara e prática:

- Propósito da ADR: Capturar decisões que afetam a estrutura, características não-funcionais, dependências, interfaces ou técnicas de construção.
- Formato e Localização: Arquivos de texto curtos (1-2 páginas), armazenados no repositório do projeto (doc/arch/adr-NNN.md), usando uma linguagem de marcação leve como Markdown.
- Numeração e Versionamento: Sequencial e monotônica. Decisões antigas são mantidas e marcadas como "superseded" (substituídas) quando revertidas, preservando o histórico.
- Estrutura de Conteúdo Padronizada:
    - Título: Nome curto e objetivo.
    - Contexto: Descrição neutra das forças em jogo (tecnológicas, políticas, sociais, do projeto).
    - Decisão: A resposta às forças, escrita em voz ativa ("Nós vamos...").
    - Status: Rastreia o estado da decisão (proposto, aceito, preterido, substituído).

- Consequências: Lista explícita de todos os resultados, positivos, negativos e neutros, da decisão.

O artigo oferece uma resposta concreta à falsa dicotomia de que projetos ágeis não podem ter uma arquitetura documentada. Ele mostra que a documentação pode ser ágil, assim como o código. A maior contribuição é a ênfase em capturar o contexto e as consequências, não apenas a decisão em si. Isso responde diretamente ao dilema do novo desenvolvedor que chega ao projeto: "O que eles estavam pensando?".

O formato é tão bem definido e simples que uma equipe pode começar a usá-lo imediatamente, sem a necessidade de ferramentas complexas. Ao armazenar as ADRs no mesmo repositório do código, elas se tornam parte do ciclo de vida do produto, sujeitas a revisão, commit e merge, assim como o código-fonte.

A eficácia do método depende do comprometimento da equipe em manter os documentos atualizados. O artigo reconhece que documentos grandes não são atualizados, mas mesmo documentos pequenos podem ser negligenciados sem disciplina cultural. O artigo aborda, mas talvez subestime, a barreira que o versionamento (como Git) pode representar para gerentes de projeto ou clientes que não estão imersos nessas ferramentas. A solução proposta (usar o GitHub para visualização) é boa, mas pressupõe que todos os stakeholders tenham acesso e familiaridade com a plataforma.

Mais de uma década após sua popularização, a prática das ADRs mostrou-se notavelmente resiliente e relevante. O artigo original capturou uma necessidade fundamental que só cresceu com a aceleração dos ciclos de desenvolvimento e a popularidade do DevOps e de microsserviços, onde decisões de interdependência e interfaces são cruciais.

A análise do artigo é extremamente perspicaz. Sua maior força está na combinação de simplicidade (o formato) com profundidade (o foco no contexto e consequências). Ele não cai na armadilha de tentar documentar tudo, mas foca no que realmente importa para a evolução saudável do sistema.