Nygard começa por enquadrar o desafio no contexto ágil. A arquitetura em projetos ágeis não é definida de uma só vez no início; ela evolui. As decisões são tomadas de forma incremental, à medida que a equipa aprende mais sobre o domínio e os requisitos. Neste cenário, a documentação de design tradicional falha espetacularmente. Documentos grandes são impossíveis de manter atualizados e, como consequência, ninguém confia neles.

O verdadeiro problema, no entanto, é a amnésia do projeto. À medida que o tempo passa e os membros da equipa mudam, a lógica por trás de decisões cruciais desaparece. Um novo desenvolvedor (ou mesmo um antigo, meses depois) depara-se com uma parte do sistema e fica perplexo, sem entender a lógica por trás de uma determinada escolha. Esta falta de contexto cria um dilema perigoso com duas saídas igualmente problemáticas: **Aceitação Cega:** O desenvolvedor assume que a decisão original ainda é válida e continua a construir sobre ela. O perigo aqui é que o contexto original (uma limitação tecnológica, um requisito de negócio, uma restrição de orçamento) pode já não existir. Ao aceitar cegamente decisões passadas, a equipa corre o risco de perpetuar soluções subótimas e de criar um sistema cada vez mais frágil, onde o medo de mudar paralisa o progresso. **Alteração Cega:** Alternativamente, o desenvolvedor, frustrado com a decisão aparentemente ilógica, decide alterá-la sem compreender totalmente as suas implicações. A decisão original pode ter sido um compromisso deliberado para suportar um requisito não funcional vital, como desempenho, segurança ou escalabilidade. Uma alteração impensada pode, inadvertidamente, comprometer a integridade do sistema, introduzindo bugs subtis ou vulnerabilidades que só se manifestarão em produção, com consequências potencialmente desastrosas.

Nygard defende que a solução para este dilema não é mais documentação, mas sim uma documentação *diferente* — uma que seja leve, focada e que preserve o conhecimento mais valioso: a justificação das decisões.

A solução proposta é a prática de manter "Registos de Decisão de Arquitetura" (Architecture Decision Records - ADRs). Em vez de tentar documentar o estado *final* da arquitetura (um alvo em constante movimento), a equipa deve documentar as *decisões* significativas que a moldam.

Cada ADR é numerado sequencialmente e foca-se numa única decisão, seguindo uma estrutura simples e concisa:

- **Título:** Uma frase curta que resume a decisão. Ex: "ADR 009: Uso de LDAP para Integração Multitenant".
- **Contexto:** Descreve as forças em jogo que levaram à necessidade da decisão. Inclui fatores tecnológicos, de negócio, sociais e políticos. Esta secção é uma descrição factual e neutra das tensões existentes.
- **Decisão:** A resposta clara e direta a essas forças. É declarada em voz ativa, por exemplo: "Nós iremos implementar um gateway de autenticação que se integra com o sistema LDAP do cliente".
- **Status:** Indica o estado da decisão: "Proposta", "Aceite" ou "Substituída" (com uma referência ao ADR que a substitui). Manter registos antigos marcados como substituídos é vital para entender a trajetória evolutiva do projeto.

- **Consequências:** Descreve o novo contexto após a aplicação da decisão. É fundamental listar todas as consequências — as boas, as más e as neutras. Por exemplo, uma decisão pode melhorar o desempenho, mas aumentar a complexidade ou introduzir uma nova dependência. Ser explícito sobre estas trocas de valor é essencial para a tomada de decisões futuras.

A força deste formato reside na sua simplicidade e foco. Um ADR é projetado para ser fácil de escrever, fácil de ler e fácil de encontrar.

A adoção da prática de ADRs tem consequências profundas e positivas para uma equipa de desenvolvimento.

Visibilidade e Onboarding: A consequência mais imediata é que a justificação por trás da arquitetura se torna explícita. Novos membros da equipa podem ler os ADRs para rapidamente ganhar contexto e compreender a história do projeto, tornando-se produtivos de forma mais rápida e segura. A questão "O que eles estavam a pensar?" deixa de ser um mistério.

Evolução Segura da Arquitetura: Os ADRs fornecem a base para uma evolução arquitetural informada. Quando a equipa considera alterar uma decisão passada, pode primeiro consultar o ADR correspondente. Se o contexto descrito no ADR já não for válido, a equipa pode tomar uma nova decisão com confiança, sabendo que não está a ignorar uma restrição importante. As consequências de um ADR frequentemente formam o contexto para o próximo, criando uma narrativa clara e rastreável.

Alinhamento da Equipa e Comunicação: O processo de escrever um ADR força a equipa a articular claramente as suas suposições, os compromissos que está a fazer e as razões para tal. Isto promove um entendimento partilhado e um alinhamento mais forte em torno das direções técnicas.

Experiência Prática e Acessibilidade: Nygard relata que a sua experiência com ADRs foi extremamente positiva, tanto com desenvolvedores como com clientes. Ele também aborda a preocupação de que manter a documentação no Git a torna inacessível para stakeholders não técnicos. Com plataformas como o GitHub, que renderizam ficheiros Markdown de forma amigável, partilhar um link para um ADR torna-se tão simples como partilhar um link para uma wiki.

Em conclusão, o artigo de Michael Nygard não defende um retorno a processos pesados de documentação. Pelo contrário, oferece uma abordagem pragmática e profundamente ágil para resolver um dos problemas mais persistentes no desenvolvimento de software. Os Registos de Decisão de Arquitetura são uma ferramenta de baixo custo e alto impacto para preservar o conhecimento mais crítico de um projeto. Ao focar-se no "porquê" em vez de apenas no "o quê", as equipas podem construir sistemas mais sustentáveis, tomar decisões melhores e, finalmente, entregar mais valor de forma consistente ao longo do tempo.